ORSAY
N° d'ordre : **2522**

THÈSE

présentée à

L'UNIVERSITÉ DE PARIS-SUD
CENTRE D'ORSAY

pour obtenir

LE GRADE DE DOCTEUR D'ÉTAT

par

Nicolas SPYRATOS

Spécialité : **Informatique**

Sujet de la thèse :

APPROCHE OPÉRATIONNELLE DES BASES DE DONNÉES

Soutenue le 23 Octobre 1981 devant la Commission composée de :

MM.	**E.**	**GELENBE**	Président
	C.	**CHABBERT**	
	C.	**DELOBEL**	
	B.	**LORHO**	Examinateurs
	C.	**PAPADIMITRIOU**	
	H.	**WEBER**	

ORSAY
N° d'ordre : **2522**

THÈSE

présentée à

L'UNIVERSITÉ DE PARIS-SUD
CENTRE D'ORSAY

pour obtenir

LE GRADE DE DOCTEUR D'ÉTAT

par

Nicolas SPYRATOS

Spécialité : **Informatique**

Sujet de la thèse :

APPROCHE OPÉRATIONNELLE DES BASES DE DONNÉES

Soutenue le 23 Octobre 1981 devant la Commission composée de :

MM.	**E.**	**GELENBE**	Président
	C.	**CHABBERT**	
	C.	**DELOBEL**	
	B.	**LORHO**	Examinateurs
	C.	**PAPADIMITRIOU**	
	H.	**WEBER**	

UNIV.
LEIDEN
BIBL.

Je remercie,

Monsieur le Professeur GELENBE qui a accepté de présider le jury de cette thèse et m'a beaucoup encouragé tout au long de ce travail,

Monsieur le Professeur DELOBEL qui m'a fait l'honneur d'être l'un des rapporteurs,

Monsieur le Professeur LORHO pour son encouragement constant et son amitié,

Messieurs les Professeurs CHABERT et WEBER qui m'ont fait l'honneur de participer au jury,

Τόν φίλο μου Χρῖστο ΠΑΠΑΔΗΜΗΤΡΙΟΥ πού λέει παρών στά προσκλητήρια τῶν καιρῶν,

Serge ABITEBOUL pour les discussions fructueuses et amicales que nous avons échangées ensemble,

Mon ami François BANCILHON pour son sens de l'absurde, ingrédient indispensable d'une recherche créative.

Je remercie également,

Mesdames Martine CORNELIS et Marie-Pierre RUBIN pour la frappe de cet ouvrage.

Ὦ ξεῖν' ἀγγέλλειν Ἕλλησι
ὅτι τῇδε κείμεθα
τοῖς κείνων ῥήμασι πειθόμενοι

Παρίσι 1981

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41547326_0036

TABLE DE MATIERES

	pages
INTRODUCTION	2
1. LA DEFINITION D'UNE BASE DE DONNEES	7
1.1. L'approche classique	8
- Variable de base relationnelle	9
- Opérations sur les relations	10
- Contraintes d'intégrité	12
- Transformations des bases	13
1.2. L'approche opérationnelle : un exemple	18
2. SPECIFICATION OPERATIONNELLE ET REALISATION DES VUES	27
2.1. Rappel de quelques notions mathématiques	28
2.2. Définition des objets	32
2.3. Définition des opérations	41
2.4. Traduction des opérations	48
2.5. Réalisation d'une vue	53
3. APPLICATIONS	55
3.1. La décomposition d'une base	56
- Définition d'une décomposition	57
- Structure de l'ensemble des vues	62
- Contenu d'une décomposition	65
3.2. Traduction dans une décomposition	68
- Gestion des vues	68
- Indépendance	71
- Formes ovales	78
4. CONCLUSION	83
- Protection de l'information	85
- Machines abstraites et bases de données	86
- Opérations et contraintes d'intégrité	87

INTRODUCTION

Une base de données est un ensemble d'informations mémorisées physiquement, qui représentent un certain nombre de faits du monde réel. Ces informations sont périodiquement consultées par les utilisateurs de la base, et mises à jour au fur et à mesure que les faits représentés changent.

On pourrait donc considérer une base comme une variable, disons BASE, dont la valeur à chaque instant représente l'ensemble des informations codées dans la base. La définition de cette variable et des opérations sur l'ensemble de ses valeurs (consultations, mises à jour) se fait à partir d'une structure élémentaire, telle que la relation, l'arbre, le graphe etc. Le choix d'une structure détermine ce qu'il est convenu d'appeler dans la littérature un "modèle de données". Ainsi, la définition de BASE à partir de la relation comme structure élémentaire et des opérations sur des relations, donne lieu au "modèle relationnel".

Le choix d'une structure élémentaire implique un "type" de valeurs pour BASE, c'est-à-dire un ensemble précis de valeurs, disons T. Néanmoins certaines valeurs de T sont souvent exclues. En effet, les faits du monde réel représentés dans une base de données obéissent en général à un certain nombre de propriétés, par exemple "tout être humain est un homme ou une femme" ou "un employé ne peut avoir qu'un seul salaire". La transposition de ces propriétés du monde réel à la base de données définit un ensemble de "contraintes d'intégrité", qui sont les règles que doit respecter la base pour être une représentation fidèle de la réalité. Ces contraintes définissent donc un ensemble, disons S, de valeurs possibles pour BASE. De façon plus formelle, l'ensemble des contraintes d'intégrité définit un prédicat C sur T. Bien évidemment, pour qu'une opération (mise à jour) u soit permise sur BASE, elle doit transformer la valeur courante de BASE en une valeur de T qui satisfasse C.

Pour résumer, la réalisation d'une base de données fait appel directement à

1. une variable BASE,
2. un ensemble de valeurs (type) T,
3. un ensemble de contraintes d'intégrité (prédicats) C sur T

et indirectement à

4. un ensemble d'opérations (mises à jour) U sur BASE

Les utilisateurs d'une base de données n'ont pas tous les mêmes besoins. C'est-à-dire que

- l'utilisateur peut ne pas avoir besoin de l'ensemble des informations de la base,
- l'utilisateur peut vouloir structurer ses données différement de la façon dont elles le sont dans la base.

Ceci conduit à la définition d'une "vue" de la base. Une vue est une nouvelle base liée à la base de départ. En effet sa définition se fait par l'introduction d'une nouvelle variable VUE de type T'("calculable" à partir de T) et d'une application f : S→T'. La valeur courante de VUE est f(s), où s est la valeur courante de BASE. Evidemment on consulte et on met à jour une vue au même titre que la base de départ. Ceci nécessite un mécanisme de "traduction" des opérations sur la vue en opérations sur la base de départ, et vice versa. Les problèmes présentés par cette traduction deviennent encore plus complexes dans une "décomposition", c'est-à-dire lorsque plusieurs vues entrent en jeu, vues définies sur la même base ("problèmes d'interférence").

Pour résumer, la définition de chaque vue fait appel directement à

5. une variable VUE,
6. un ensemble de valeurs (type) T', lié à T,

7. une application f : S→T'

et indirectement à

8. un ensemble d'opérations (mises à jour) U' sur VUE, lié à U
   et aux ensembles d'opérations d'autres vues, à travers l'application f.

Etant donnée l'importance de la traduction des opérations dans une décomposition, plusieurs aspects de ce problème, tels que l'indépendance, les formes normales etc., ont été étudiés dans la littérature. Ces études s'appuient sur un modèle, c'est-à-dire qu'elles font appel aux concepts 1 à 8 cités plus haut. Cette multitude de concepts impliqués cache, à notre avis, la vraie nature du problème qui est celle de la transformation de l'information contenue dans la base,

Nous estimons que le problème de traduction peut être énoncé et, en grande partie, étudié en dehors d'un modèle, en s' appuyant uniquement sur le concept de l'opération qui est le seul concept directement impliqué dans la traduction. C'est la raison pour laquelle nous proposons dans cette thèse une approche entièrement opérationnelle des bases de données. Cette approche permet :

- une définition de la base se rapprochant des types abstraits de données ,
- l'étude du contenu de l'information d'une base sans faire appel à un modèle spécifique,
- le traitement, d'une manière simple et uniforme, des principaux concepts relatifs aux bases de données.

Bien entendu la possibilité de représenter les concepts impliqués suivant un modèle n'est pas négligée. Elle est tout simplement séparée de la spécification opérationnelle. En effet, l'approche proposée considère la définition d'une vue à deux niveaux :

Spécification opérationnelle : La vue est spécifiée par un couple (D,O) d'ensembles d'opérations sur la base de départ tels que

- D induit une partition π de l'ensemble des valeurs S de la base de départ et ceci définit les objets de la vue
- O induit des opérations sur les objets de la vue, c'est à dire que chaque opération de O peut être traduite en une opération sur la vue.

Réalisation : La vue est réalisée par un couple (π,π') de deux partitions telles que

- π est la partition de S induite par D
- π' est la partition de O qui regroupe dans chaque bloc toutes les opérations qui ont la même traduction sur la vue.

Ce sont les blocs de ces deux partitions qui sont éventuellement "nommés", suivant un modèle, pour réaliser la vue.

Notons que la spécification opérationnelle place toutes les vues sur le même plan, en ce sens qu'elles sont toutes définies à partir des opérations sur la base de départ.

Notons aussi qu' à cette décomposition horizontale, en spécification opérationnelle et en réalisation, correspond une décomposition verticale, en partie statique et en partie dynamique. En effet, l'ensemble D et la partition π représentent le contenu de l'information de la vue, alors que l'ensemble O et la partition π' représentent la possibilité de transformer cette information.

Ceci résume l'approche opérationnelle des bases de données, idée développée à partir d'un travail présenté dans [27]. Cette approche peut être appliquée à différents concepts des bases de données comme l'indique l'organisation de la thèse ci-dessous.

CHAPITRE 1 : Nous résumons l'approche classique des bases de données à travers le modèle relationnel, et nous présentons un exemple d'introduction à l'approche proposée.

CHAPITRE 2 : Nous présentons la définition formelle d'une vue en deux niveaux

- spécification opérationnelle
- réalisation

Comme cette approche ne fait appel à un modèle qu'au niveau de la réalisation, elle se rapproche des types abstraits de données,

CHAPITRE 3 : Nous présentons une étude de la décomposition d'une base par l'approche opérationnelle. En particulier, nous définissons une structure de l'ensemble des décompositions d'une base qui nous permet de

- comparer deux décompositions
- définir leur équivalence.

Les résultats de cette étude sont ensuite appliqués à trois problèmes importants des bases de données

- la gestion des vues d'une décomposition
- l'indépendance
- la définition d'une "vue première", notion en relation avec les formes normales.

CHAPITRE 4 : En conclusion, nous esquissons une application possible de l'approche opérationnelle à la protection de l'information, et nous discutons des parallèles qui existent entre machines abstraites et bases de données. Nous proposons enfin une représentation possible des contraintes d'intégrité par des opérations.

CHAPITRE 1. LA DEFINITION D'UNE BASE DE DONNEES.

Dans ce chapitre, nous résumons l'approche classique des bases de données à travers le modèle relationnel, et nous présentons un exemple d'introduction à une approche opérationnelle.

1.1. L'APPROCHE CLASSIQUE

Une base de données est un ensemble d'informations mémorisées physiquement, qui représentent un certain nombre de faits du monde réel. Ces informations sont périodiquement consultées par les utilisateurs de la base, et mises à jour au fur et à mesure que les faits représentés changent. Par exemple, on pourrait imaginer que les informations concernant les employés d'une entreprise, leurs salaires, et leur lieu de travail constituent une base de données. Une telle base serait consultée par le service de traitement pour connaître la liste des employés travaillant à un endroit donné, et mise à jour par le service du personnel après l'embauche d'un nouvel employé.

On pourrait donc considérer une base comme une variable, disons BASE, dont la valeur à chaque instant représente l'ensemble des informations codées dans la base. Une consultation de la base correspond alors à une "visualisation" de tout ou "partie" de la valeur courante de BASE, et une mise-à-jour de la base correspond à un changement de valeur de BASE. L'ensemble des consultations et des mises-à-jour que l'on peut effectuer sur la base est appelé un "langage de manipulations". En particulier, l'ensemble de consultations est appelé souvent un "langage de requêtes".

La définition formelle de la variable BASE et des opérations sur l'ensemble de ses valeurs (consultations, mises-à-jour) se fait à partir d'une structure élémentaire (telle que la relation, l'arbre, le graphe ou autre structure formelle) et des opérations sur cette structure. A titre d'exemple,on verra ici la définition formelle d'une base de données à partir de la relation, comme structure élémentaire, et des opérations sur des relations. Cette façon de définir une base de données est appelée dans la littérature le "modèle relationnel".

Variable de base relationnelle

Soit $A = \{A_1, A_2, \ldots, A_n\}$ un ensemble de noms, que l'on appelera des attributs. Un ensemble de valeurs T_i est associé à chaque attribut A_i, $i=1,2,\ldots,n$. L'ensemble T_i est appelé le type de A_i. Un n-uplet sur A est une application qui associe à chaque attribut A_i de A une valeur de T_i. Une relation sur A est un ensemble quelconque de n-uplets sur A. Une variable relationnelle V sur A est une variable dont les valeurs sont des relations sur A. Une telle variable pourrait donc être définie en PASCAL de la façon suivante :

type TUPLE = record A_i:T_1; A_2:T_2; ...; A_n:T_n end

type RELATION = set of TUPLE

var V : RELATION

D'une façon informelle, on utilisera la notation

$V(A_1:T_1; A_2:T_2; \ldots; A_n:T_n)$

appelée dans la littérature schéma de relation, pour indiquer que V est une variable dont les valeurs sont des relations sur A. Voici un exemple de définition d'une variable relationnelle :

type NOM = packed array [1..20] of CHAR

type TUPLE = record EMPLOYE : NOM ; SALAIRE : INTEGER end

type RELATION = set of TUPLE

var PAYE : RELATION

Les attributs sont EMPLOYE et SALAIRE. Les types correspondant sont NOM et INTEGER. La variable relationnelle est PAYE. Le schéma de relation qui résume, en quelque sorte, la définition de PAYE est le suivant :

PAYE(EMPLOYE:NOM; SALAIRE:INTEGER)

Suivant un abus de notation très répandu dans la littérature, on notera plutôt :

PAYE(EMPLOYE,SALAIRE)

les ensembles de valeurs étant implicites. Ainsi, dans le schéma
SITE(EMPLOYE,MAGASIN)
il est entendu que le type de MAGASIN est un ensemble de noms de magasins.

Une variable de base relationnelle est un ensemble ordonné de variables relationnelles, que l'on notera entre parenthèses. Par exemple, on pourrait définir une variable de base R en se servant des variables relationnelles PAYE et SITE définies plus haut, soit

R = (PAYE,SITE)

De toute affectation de valeurs aux variables relationnelles résulte une valeur de la variable de base. L'ensemble de toutes les valeurs de la variable de base est son type. Par exemple, si T1 et T2 sont les types de variables relationnelles PAYE et SITE, le type de R est le produit cartesien, au sens mathématique, de leurs ensembles des parties : $T_1 \times T_2$. Lorsque la variable de base consiste en une seule variable relationnelle V, on notera V, au lieu de (V), pour la variable de base.

On appelera une base de données, ou tout simplement une base, tout couple B=(R,S), où R est une variable de base de type T et S un sous ensemble de T. On appellera S l'ensemble de valeurs de B .

Opérations sur les relations

Nous définissons ici quelques opérations sur les relations dont nous aurons besoin plus tard. Ce faisant, nous utiliserons le symbole d'une variable relationnelle pour indiquer aussi sa valeur

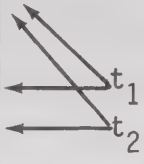
courante. Ainsi, si la valeur courante de PAYE contient uniquement les deux n-uplets suivants :

t_1 = {(EMPLOYE, Dupont), (SALAIRE, 7000)}

t_2 = {(EMPLOYE, Martin), (SALAIRE, 6000)}

on écrira PAYE= $\{t_1, t_2\}$ et on dira "la relation PAYE" au lieu "variable PAYE". En général, on représente la valeur d'une variable relationnelle sour la forme d'un tableau comme l'indique l'exemple ci-dessous :

EMPLOYE	SALAIRE
Dupont	7000
Martin	6000



On utilisera le symbole t.X pour indiquer la restriction d'un n-uplet t à un sous-ensemble X de ses attributs. Ainsi, dans l'exemple précédent, nous avons :

t_1.{EMPLOYE} = {(EMPLOYE,Dupont)}

Soit V et W deux relations sur des ensembles d'attributs correspondant A et B. La projection de V sur un sous-ensemble X de A, noteé V[X], est une relation sur X definie comme suit

$V[X] = \{t.X | t \in V\}$

Soit ℓ un n-uplet sur $A \cup B$. La jointure de V et W, notée V*W, est une relation sur $A \cup B$ définie comme suit

$V*W = \{\ell | \ell.A \in V, \ell.B \in W\}$

Soit C une condition(prédicat)sur l'ensemble de n-uplets sur A. La sélection de V par rapport à C, notée (V|C), est une relation sur A définie comme suit

$(V|C) = \{t \in V | C(t) = vrai\}$

Contraintes d'intégrité

Les faits du monde réel représentés dans une base de données obéissent en général à un certain nombre de propriétés concernant leur sémantique. Par exemple, "chaque être humain est un homme ou une femme", ou "un employé ne peut avoir qu'un seul salaire". La transposition de ces propriétés du monde réel à la base de données définit un ensemble de contraintes d'intégrité, qui sont les règles que doit respecter la base pour être une représentation fidèle de la réalité. Ces contraintes définissent donc un ensemble de valeurs possibles de la base. De façon plus formelle, étant donnée une variable de base R, de type T, l'ensemble des contraintes d'intégrité sur R définit un prédicat C sur T et par conséquent restreint la base de données à : B = (R,S) tel que S = $\{R \in T | C(R) = vrai\}$.

Voici un exemple de définition d'une base de données par des contraintes d'intégrité. Soit la variable de base

R = (PAYE,SITE)

définie plus haut. Le fait que "chaque salarié de PAYE doit travailler dans un magasin de SITE" est une contrainte d'intégrité C1 qui peut être exprimée par des opérations sur les relations PAYE et SITE :

C1 : PAYE[EMPLOYE] $\subseteq$ SITE[EMPLOYE]

Cette contrainte définit la base suivante sur R

B = (R,S1) tel que S1 = $\{R \in T | C1(R) = vrai\}$

Un ensemble de contraintes d'intégrité particulièrement intéressant et qui a fait l'objet de nombreuses études [3,29] est celui des dépendances fonctionnelles. Soit V une variable relationnelle sur un ensemble d'attributs A. Soit X et Y deux sous-ensembles de A. On dit que la relation V vérifie la dépendance fonctionnelle (X,Y) notée "X $\rightarrow$ Y dans V", ssi $\forall t \in V, \forall u \in V,\ t.X = u.X \Rightarrow t.Y = u.Y$

Prenons de nouveau l'exemple de la variable de base

R = (PAYE,SITE)

Le fait qu'"un employé doit être affecté à un magasin et à un seul" s'exprime par dépendance fonctionnelle

C2 : EMPLOYE → MAGASIN dans SITE

Le couple

B = (R,S) tel que $S = \{R \in T | C(R) = vrai\}$, où $C = C1 \wedge C2$
est une nouvelle base sur R et on a, bien évidemment, $S \subseteq S1$.

A noter que l'absence de toute contrainte sur une variable de base R correspond à la base (R,T).

Bien que les contraintes d'intégrité fournissent un moyen naturel de définition des bases de données, il en existe d'autres. Notamment la transformation de l'ensemble des valeurs d'une variable donnée, que l'on étudiera par la suite, conduit à la création de nouvelles bases-images à partir d'une base donnée.

Transformations des bases

Les informations emmagasinées dans une base sont périodiquement consultées et mises à jour par les utilisateurs. Dans les deux cas, la base de données est "transformée". Prenons un exemple. Soit deux variables relationnelles D et E définies par les schémas suivants :

D(RAYON, DIRECTEUR)
E(EMPLOYE , RAYON)

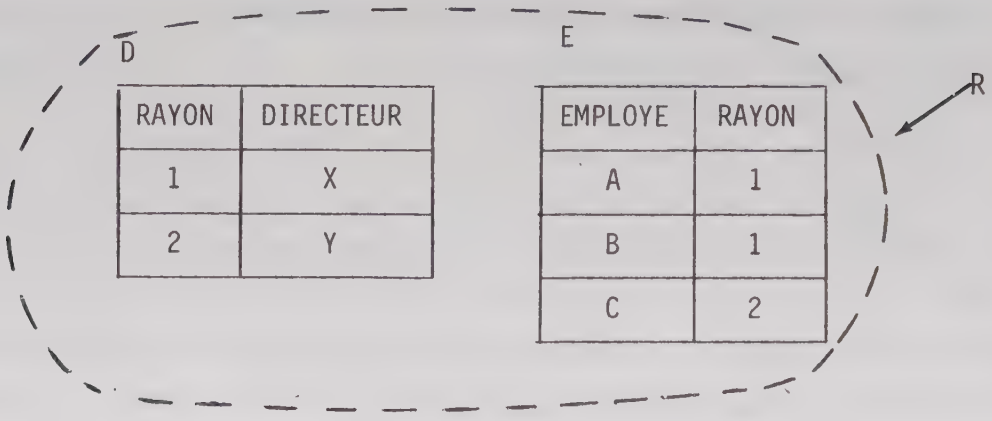
Soit une variable de base :

R = (D,E) de type T
et les contraintes d'intégrité
C1 : RAYON → DIRECTEUR dans D
C2 : EMPLOYE → RAYON dans E
C3 : D[RAYON] = E [RAYON]

Soit la base de données :

B = (R,S) tel que $S = \{R \in T | C(R) = vrai\}$, où $C = C1 \wedge C2 \wedge C3$

Ci-dessous un exemple de valeur de cette base :



D

RAYON	DIRECTEUR
1	X
2	Y

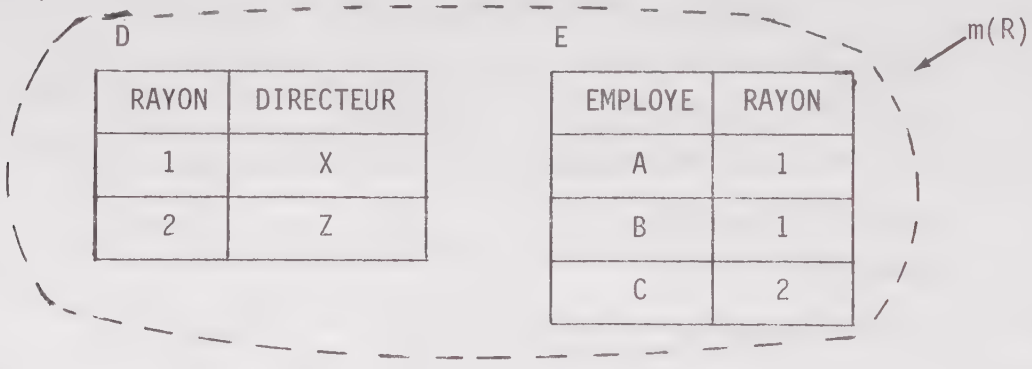
E

EMPLOYE	RAYON
A	1
B	1
C	2

Un exemple de mise à jour sur cette base serait :

m : Remplacer le directeur du rayon 2 par Z (en supposant que le rayon 2 existe dans la valeur courante de la base).

Cette mise à jour, appliquée à la valeur courante R, la "transforme" en la nouvelle valeur m(R), ci-dessous, qui satisfait les contraintes C1, C2, et C3, énoncées plus haut.



D

RAYON	DIRECTEUR
1	X
2	Z

E

EMPLOYE	RAYON
A	1
B	1
C	2

Comme la valeur courante R de la base peut être n'importe quelle valeur satisfaisant les contraintes C1, C2, et C3, on conclut que m est bien une application de l'ensemble de valeurs S dans lui-même.
D'autre part la mise à jour :

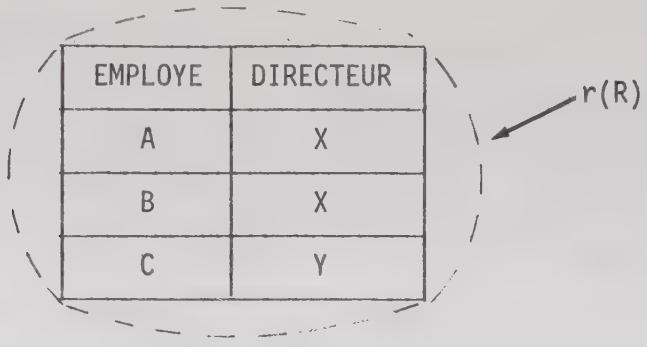
m' : Ajouter l'employé A au rayon 2 n'est pas une application de S dans S, puisque $m'(R) \notin S$.

Dans ce travail nous ne considérons que des mises à jour qui sont des applications de S dans S.

Un exemple de consultation ou "requête" sur cette même base est le suivant :

r : Lister tous les employés et leurs directeurs.

Le traitement de cette requête nécessite la jointure de deux relations D et E, suivie par une projection sur l'ensemble des attributs {EMPLOYE,DIRECTEUR}. Le résultat de ce traitement, sur la valeur courante R de la base, est la relation suivante :



Notons que cette relation est "calculée" à partir de relations de la base, mais qu'elle ne figure pas elle même dans la base. On pourrait néanmoins nommer cette relation, disons R', c'est à dire qu'on pourrait définir une nouvelle variable relationnelle R', de type T', par le schéma suivant :

R'(EMPLOYE, DIRECTEUR)

et convenir que la valeur courante de R' s'obtient par l'affectation de r(R) à R'. Ainsi la consultation r devient une application

$r : S \rightarrow T'$ tel que $r(R) = (D*E)[EMPLOYE,DIRECTEUR]$

Cette application "transforme", en effet, la base de départ

$B = (R,S)$ tel que $S = \{R \in T \mid C(R) = vrai\}$

en une nouvelle base-image

$B' = (R',S')$ tel que $S' = r(S)$

Notons que l'ensemble des valeurs de B' peut, dans ce cas, être défini par la contrainte

C' : EMPLOYE → DIRECTEUR dans R'

c'est-à-dire que l'on a

$S' = r(S) = \{R' \in T' \mid C'(R') = vrai\}$

Le procédé suivi plus haut pour la définition d'une nouvelle base B', à partir d'une base B, est précisément celui qui est suivi pour définir une vue. Etant donnée une base B = (R,S), une vue de B est une base B' = (R',S') dont l'ensemble de valeurs S' est l'image d'une application r : S → T', c'est-à-dire, S' = r(S). Ceci suppose évidemment que le type T' soit "calculable" à partir de T.

La notion de vue est aussi connue sous les noms schéma externe et sous schéma utilisateur [2]. L'introduction d'un mécanisme de définition des vues dans les systèmes actuels de gestion des bases de données [12] est due, surtout, au fait que des applications particulières ont des besoins particuliers. Ainsi dans le cadre d'une application précise :

1. L'utilisateur peut ne pas avoir besoin de l'ensemble des données de la base (c'est-à-dire que certains attributs, relations ou parties de relations ne lui sont pas nécessaires) et qu'il est donc inutile de l'encombrer de la description de l'ensemble des données.

2. L'utilisateur peut vouloir structurer ses données différement de la façon dont elles le sont dans la base.

A travers les exemples étudiés jusqu'ici,on s'aperçoit que la notion d'application, un outil mathématique à la fois simple et puissant, est l'ingrédient principal de toute transformation d'une base, En effet étant donnée une base B = (R,S), on définit une transformation de B comme tout couple (R',f) ou R' est une variable de base, de type T', et f une application de S dans T'. L'image de B par la transformation (R',f) est la base B' = (R',f(S)). On a déjà vu que cette définition d'une transformation s'applique à la consultation et à la mise à jour d'une base ainsi qu'à la définition des vues. Mais on pourrait citer d'autres cas, notamment le processus de décomposition d'une relation, la répartition d'une base et le processus d'abstraction.

Cette diversité d'applications des transformations d'une base nous a conduit à étudier la structure de l'ensemble de ces transformations en termes très généraux. Les résultats de cette étude ont été ensuite appliqués à des cas particuliers :

- processus de décomposition [5]
- mise à jour des vues [4]
- processus d'abstraction [28]

Ce que l'on retiendra, après ce survol rapide de l'approche classique des bases de données, c'est que la transformation d'une base est un outil général qui nous permet de définir de nouvelles bases (vues) à partir d'une base donnée. Mais alors qu' une transformation de base nous conduit vers l' "extérieur" de la base (c'est-à-dire, qu'elle nécessite la définition de nouvelles variables de bases et de nouveaux types), l'approche opérationnelle que l'on étudiera par la suite nous permet de traiter les mêmes problèmes, d'une façon plus simple, par des transformations vers l' "intérieur" de la base (c'est-à-dire par des applications de l'ensemble des valeurs dans lui-même).

1.2. L'APPROCHE OPERATIONNELLE : UN EXEMPLE

Une base de données a été définie comme un couple (R,S), où R est une variable de base et S un sous-ensemble du type T de R. Cette définition "classique" est liée aux valeurs du type T et met en jeu un nombre important de concepts, en particulier :

1. le type T
2. la variable de base R
3. l'ensemble des valeurs S
4. un prédicat C sur T (pour la définition de S)

et pour chaque transformation de base (R',f),

5. un nouveau type T'
6. une nouvelle variable R'
7. une application f : S → T'

Dans ce travail, on ne s'intéresse pas aux valeurs de S mais uniquement à son comportement. Ce comportement est détecté à travers des opérations permises sur S, c'est-à-dire, à travers des applications de S dans lui-même. De ce point de vue, deux valeurs (que l'on ne "voit" pas) se comportent de la même façon, ou sont "équivalentes", si elles ne peuvent pas être distinguées par des opérations sur S. Nous proposons donc une nouvelle définition "opérationnelle" d'une vue, en nous servant uniquement des opérations sur S. Cette définition présente de multiples avantages :

- elle utilise explicitement le concept d'opération sur S, notion directement impliquée dans des problèmes importants de bases de données, tels que la gestion de vues, les formes normales etc.
- elle n'utilise qu'un seul concept et elle est de ce fait beaucoup plus facile à manipuler
- elle n'est pas liée à un modèle de données et se rapproche donc de la définition d'un "type abstrait de données".

Dans ce paragraphe nous montrerons, par des exemples, comment on peut décrire tous les concepts impliqués dans la définition classique (1 à 7 ci-dessus) uniquement par des opérations sur S. Ensuite le Chapitre 2 sera consacré à un traitement formel de cette nouvelle approche et le Chapitre 3 aux applications à divers problèmes des bases de données.

Commençons par un exemple. Soit une variable relationnelle R définie par le schéma suivant :

R(EMPLOYE,RAYON)

où le type de EMPLOYE est T1 = {A,B} et le type de RAYON est T2 = {x,y}. Soit maintenant la base (R,S) où l'ensemble des valeurs S consiste en neuf relations énumérées dans la Figure 1(a). Pour des raisons de simplicité d'écriture on note Ax au lieu de (A,x) etc... Considérons ensuite une transformation (R',f) de cette base, où R' est une variable relationnelle définie par le schéma

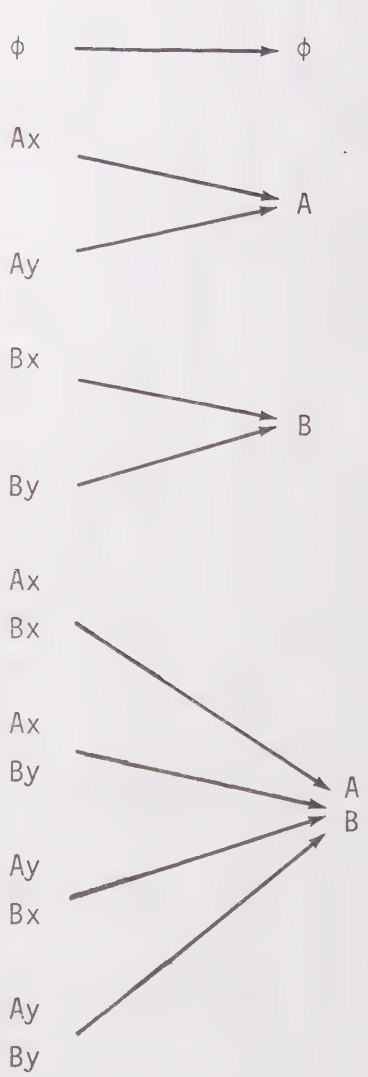
R'(EMPLOYE)

et f est une application de S dans T' définie par

f(R) = R[EMPLOYE]

Cette application est représentée dans la Figure 1(a). Il en résulte que l'image de (R,S) sous la transformation (R',f) est la base (R',S'), où S' = {ϕ,{A},{B},{A,B}}.

L'application f définit une partition de S, que l'on notera π. Cette partition est représentée à part, Figure 1(b), où des chiffres ont été utilisés à la place des valeurs de S (par exemple,1 représente ϕ, 3 représente Ay, etc..). Chaque bloc de cette partition correspond à une valeur de S', et vice versa, si bien que l'on pourrait imaginer que les blocs de π sont effectivement les valeurs de S'. Ainsi l'image (R',S') pourrait être remplacée par (R',π). Pour atteindre notre but,il ne nous reste qu'à remplacer π par un ensemble d'opérations sur S.



R	p(R)	q(R)	r(R)
1	1	1	1
2	3	2	3
3	2	3	2
4	4	5	5
5	5	4	4
6	8	7	9
7	9	6	8
8	6	9	7
9	7	8	6

(c)

Ensemble U d'opérations permises sur S

i : Ne rien changer (identité sur S)
p : Changer A de rayon
q : Changer B de rayon
r : Changer A et B de rayon
k : Si A ne figure pas dans la base, ajouter A au rayon x
ℓ : Si A ne figure pas dans la base, ajouter A au rayon y
n : Si B ne figure pas dans la base, ajouter B au rayon x
t : Si B ne figure pas dans la base, ajouter B au rayon y
u : Supprimer le n-uplet de A
v : Supprimer le n-uplet de B
w : Supprimer le rayon x
z : Supprimer le rayon y

(d)

Figure 1 - Définition d'une vue, à partir d'une base donnée
(satisfaisant la contrainte : EMPLOYE→RAYON dans R)

Considérons l'ensemble des opérations U, énumérées Figure 1(d). Si on examine bien les quatre premières opérations (parmi lesquelles p,q et r sont représentés Figure 1(c)),on s'aperçoit qu'aucune ne relie deux valeurs de S appartenant à deux blocs différents de π. En d'autres termes, ces quatre opérations ne relient que des valeurs se trouvant dans un même bloc de π, c'est-à-dire, qu'elles ne relient que des valeurs équivalentes (équivalentes parce que, à l'intérieur d'un bloc, elles ont toutes la même liste d'employés). Donc, si l'union de ces quatre opérations est une relation d'équivalence, l'ensemble de ces opérations pourrait "remplacer" π, en ce sens qu'à partir de cet ensemble, on pourrait reconstituer la partition π de S. En effet, en réunissant les quatre opérations énumérées ci-dessous sous forme d'ensembles de couples, on obtient bien une relation d'équivalence sur S dont les classes d'équivalences sont les blocs de π.

i = {(1,1), (2,2), (3,3),(4,4), (5,5), (6,6), (7,7), (8,8), (9,9)}

p = {(1,1), (2,3), (3,2),(4,4), (5,5), (6,8), (7,9), (8,6), (9,7)}

q = {(1,1), (2,2), (3,3),(4,5), (5,4), (6,7), (7,6), (8,9), (9,8)}

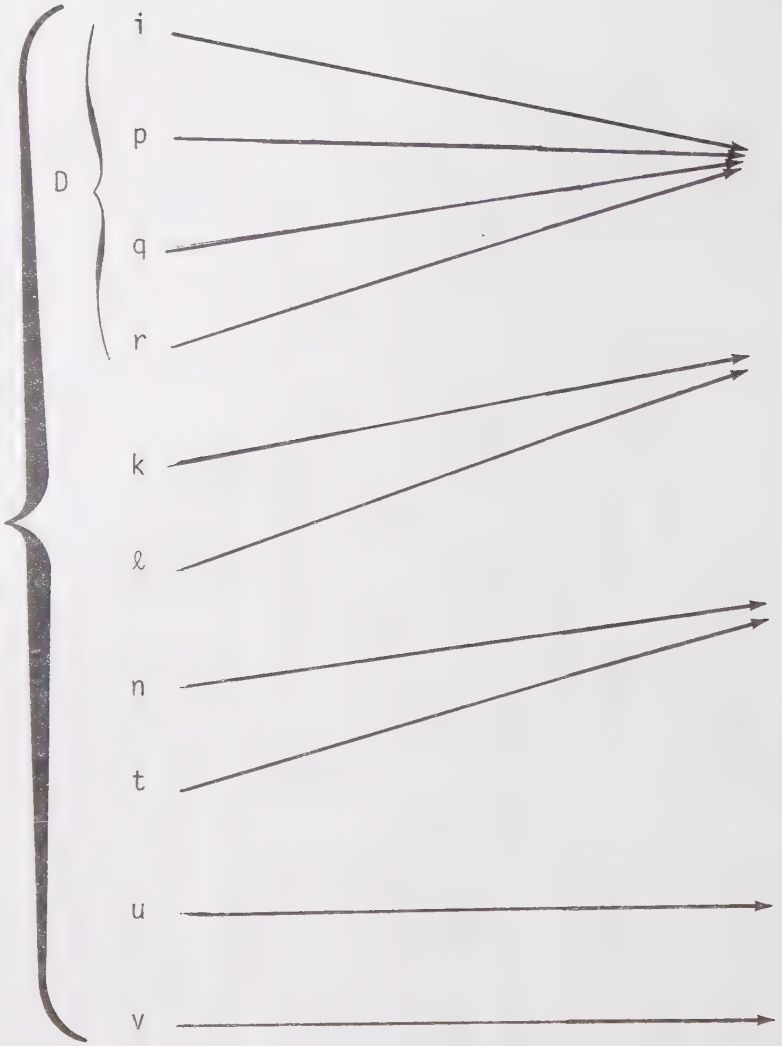
r = {(1,1), (2,3), (3,2),(4,5), (5,4), (6,9), (7,8), (8,7), (9,6)}

On peut dire donc que l'ensemble

D = {i,p,q,r}

"définit" la partition π, et ceci nous permet de remplacer le couple (R',π) par (R',D). On pourrait se demander bien sûr quelles sont les conditions nécessaires et suffisantes (exprimées en termes d'opérations) pour qu'un ensemble donné d'opérations définisse une partition. Cette question, ainsi que d'autres questions relatives à la définition par opérations, seront étudiées au chapitre suivant.

Pour résumer, nous nous sommes donnés au départ une base (R,S), un ensemble d'opérations U sur S, et une transformation (R',f). En choisissant judicieusement un sous ensemble D de U on a pu remplacer l'image (R',S') par (R',D). Maintenant, comme nous ne nous intéressons pas aux valeurs propres de R', mais seulement à leur comportement détecté à travers des opérations,



i_π : Ne rien changer (identité sur π)

a : Si A ne figure pas dans la base, ajouter A

b : Si B ne figure pas dans la base, ajouter B

c : Supprimer A

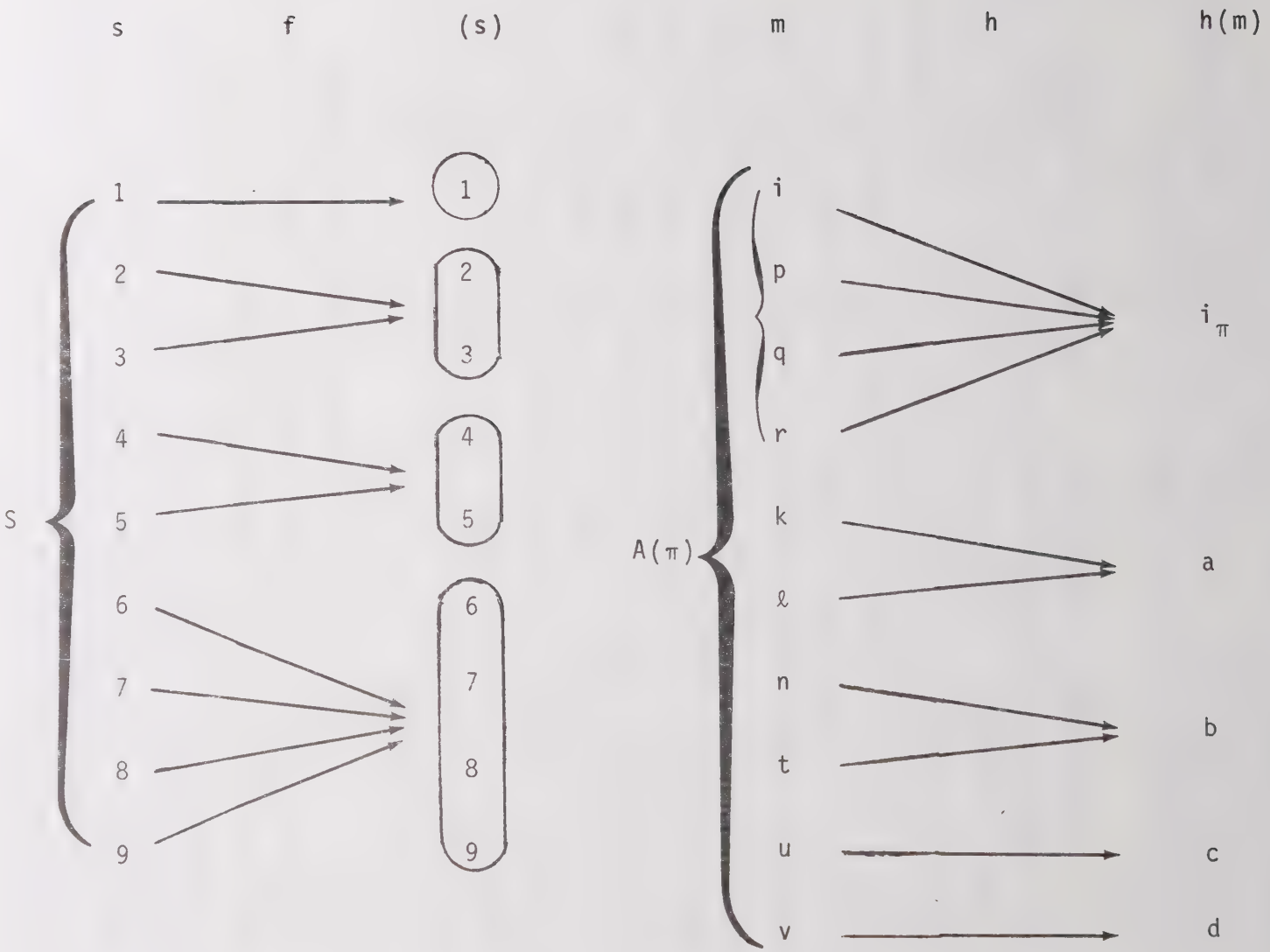
d : Supprimer B

Figure 2 - La "traduction" des opérations "applicables" en opérations sur π

on peut se passer de la variable R', c'est-à-dire que l'ensemble D remplace la transformation (R',f) et l'image (R',S'). Donc D définit une nouvelle base, que l'on appelera une vue de la base (R,S). Ceci implique qu'à partir de l'ensemble d'opérations U, on peut extraire de l'information, c'est-à-dire créer des vues, en se servant de sous-ensembles de U,tel que l'ensemble D défini plus haut.

Ayant défini une vue à partir de l'ensemble U d'opérations (voir Figure 1), on pourrait se demander quelles sont les opérations de U qui sont "applicables" à cette vue, applicables au sens qu'elles pourraient être "traduites" en opérations sur la vue. Bien évidemment, pour qu'une opération de U soit applicable sur la vue, elle doit faire passer de chaque bloc de la partition π (voir Figure 1) à un autre bloc et un seul. Parmi les opérations de U,seules les deux dernières, w et z, ne satisfont pas cette condition et ne sont donc pas applicables sur la vue. Notons que l'on aurait pu arriver à cette même conclusion d'une façon intuitive. En effet, à l'intérieur de chaque bloc, la caractéristique commune des valeurs est qu'elles ont toutes la même liste d'employés. Donc toute opération sur la base du départ, portant sur les employés,doit être applicable à la vue (qui ne contient justement que des listes d'employés). Les seules opérations qui ne portent pas sur les employés sont les deux dernières, w et z, et, de ce fait, ne sont pas applicables à la vue. On notera $A(\pi)$ l'ensemble de toutes les opérations applicables à la vue.

Il se peut que parmi les opérations applicables, deux ou plusieurs se traduisent par la même opération sur la vue. Ainsi les opérations k et ℓ (Figure 1), qui sont différentes sur la base de départ, se traduisent par la même opération a sur la vue (Figure 2). En particulier, les opérations de l'ensemble D, qui définit la vue, se traduisent toutes par l'identité. Cette traduction des opérations définit bien une application h (voir Figure 3) de l'ensemble $A(\pi)$ dans l'ensemble d'opérations possibles sur la vue. Cette application que l'on



$$\forall\, s \in S \quad \forall\, m \in A(D) \quad g(m(s)) = h(m)(g(s))$$

Figure 3 : Le processus de traduction

appelera "le traducteur", nous permet de bien représenter le processus de traduction. En effet comme on vient de l'expliquer, l'image $h(A(\pi))$ est l'ensemble des traductions de toutes les opérations applicables sur la vue. Ce sont donc les seules opérations permises sur la vue. Dans l'autre sens, étant donnée une opération e permise sur la vue, l'image inverse $h^{-1}(e)$ est l'ensemble de toutes les opérations qui permettent de "répercuter" e sur la base du départ. Ainsi, si on applique l'opération c sur la vue (voir Figure 3), il y a une façon unique de la répercuter sur la base de départ, puisque $h^{-1}(c) = \{u\}$. Par contre dans le cas de l'opération b, il y a un choix entre n et t, puisque $h^{-1}(b) = \{n,t\}$. En particulier, dans le cas de l'identité i_π, il y a un choix parmi quatre opérations, puisque $h^{-1}(i_\pi) = \{i,p,q,r\}$. Cette ambiguité du choix, pendant la "traduction inverse", peut être résolue si d'autres vues entrent en jeu (vues définies à partir de la même base). Cet aspect de la traduction sera traité formellement au Chapitre 3.

Nous pouvons aussi considérer le traducteur h comme un processus d'attribution de "noms" (dans le contexte de la vue) à des sous-ensembles de $A(\pi)$. Ainsi les opérations k et ℓ (voir Figure 1) qui sont bien différentes sur la base du départ, reçoivent un nom commun, notamment, le nom a. De même, l'ensemble D reçoit comme nom i_π, etc. D'une manière générale, h fait correspondre des noms aux blocs d'une partition, disons π', de l'ensemble $A(\pi)$. Néanmoins, c'est la partition π' qui est importante et non pas les noms qui lui sont attribués par h.

Pour mieux comprendre le comportement des opérations traduites par h, définissons une application f (voir Figure 3) qui fait correspondre à chaque valeur de S le bloc de π qui la contient. On pourrait vérifier que :

$$\forall\, s \in S \quad \forall\, m \in A(\pi) \quad f(m(s)) = h(m)(f(s))$$

ce qui montre bien que la vue munie des opérations de $A(\pi)$ traduites par h, "simule" d'une façon grossière la base de départ.

Bien évidemment, en définissant la vue, on pourrait "interdire" certaines opérations applicables, c'est-à-dire, que l'on pourrait se restreindre à un sous ensemble O de l'ensemble $A(\pi)$. Ainsi la "spécification" opérationnelle d'une vue se fait par un couple (D,O) de sous-ensembles de U tels que :

D induit les "objets", c'est-à-dire les blocs de π
O induit les opérations sur ces objets

La "réalisation" d'une telle spécification se fait en donnant des "noms" aux objets et aux opérations induits, suivant un modèle particulier. Les objets induits par D, comme on l'a vu plus haut, sont les blocs de la partition π induite par D. Quant aux opérations induites par O, ce sont les blocs de la partition π' induite par h sur O. Ainsi le couple de partitions (π,π') devient la "réalisation" de la spécification (D,O). Cette définition d'une vue en deux niveaux :

(1) spécification opérationnelle
(2) réalisation

fait l'objet d'une étude formelle au chapitre suivant.

CHAPITRE 2. SPECIFICATION OPERATIONNELLE ET REALISATION DE VUES

Dans ce chapitre nous proposons la définition d'une vue en deux niveaux :

- spécification opérationnelle
- réalisation

Cette approche ne fait appel à un modèle qu'au niveau de la réalisation et se rapproche de ce fait des types abstraits de données.

2.1. RAPPEL DE QUELQUES NOTIONS MATHEMATIQUES

Dans ce paragraphe nous rappelons quelques concepts mathématiques nécessaires pour la suite. Nous supposons bien connues les notions de partition, de relation d'équivalence et d'ordre partiel sur un ensemble.

Si π est une partition d'un ensemble S, on note $B_\pi(s)$ le bloc de π qui contient s. On se servira de barres plutôt que d'accolades pour noter les blocs. Par exemple, si $S = \{1,2,3,4,5\}$ et qu'une partition π a comme blocs $\{1,2\}$, $\{3\}$, et $\{4,5\}$, on notera $\pi = \{\overline{1,2}\ ;\ \overline{3}\ ;\ \overline{4,5}\}$. On notera $s \equiv t(\pi)$ ssi s et t appartiennent au même bloc de π , c'est-à-dire ssi $B_\pi(s) = B_\pi(t)$.

Si R est une relation d'équivalence sur S, l'ensemble de classes d'équivalences de R définit une partition π de S, et inversement, toute partition π de S définit une relation d'équivalence sur S dont les classes d'équivalence sont les blocs de π . Ainsi, si R définit π , sRt ssi $s \equiv t(\pi)$ ssi $B_\pi(s) = B_\pi(t)$.

Le "produit" de deux partitions π_1 et π_2 de S est défini comme suit :
$\pi_1.\pi_2$ est la partition sur S telle que : $s \equiv t(\pi_1.\pi_2)$ ssi $s \equiv t(\pi_1)$ et $s \equiv t(\pi_2)$.

Le calcul de $\pi_1.\pi_2$ est très simple, puisque les blocs de $\pi_1.\pi_2$ sont obtenus par intersection de blocs de π_1 et de π_2 :

$B_{\pi_1 . \pi_2}(s) = B_{\pi_1}(s) \cap B_{\pi_2}(s)$

Prenons un exemple . Soit :

$S = \{1,2,3,4,5,6,7,8,9\}$, $\pi_1 = \{\overline{1,2} ; \overline{3,4} ; \overline{5,6} ; \overline{7,8;9}\}$,
$\pi_2 = \{1,6 ; 2,3 ; 4,5 ; 7,8 ; 9\}$

En appliquant la définition du produit, on obtient :

$\pi_1 \cdot \pi_2 = \{\overline{1} ; \overline{2} ; \overline{3} ; \overline{4} ; \overline{5} ; \overline{6} ; \overline{7,8} ; \overline{9}\}$

Etant données deux partitions π_1 et π_2 de S, on dit que π_1 est inférieure ou égale à π_2, noté $\pi_1 \leq \pi_2$, ssi chaque bloc de π_1 est contenu dans un bloc de π_2. Donc " $\pi_1 \leq \pi_2$" doit être interprété"π_1 est plus fin que π_2". Il en résulte que :

$\pi_1 \leq \pi_2$ ssi $\pi_1 \cdot \pi_2 = \pi_1$

Si π et τ sont des partitions sur S et $\pi \geq \tau$, π définit une partition $\overline{\pi}$ de l'ensemble des blocs de π comme suit :

$B_\tau(s) \equiv B_\tau(t)(\overline{\pi})$ ssi $s \equiv t(\pi)$

Ainsi deux blocs de τ appartiennent au même bloc de $\overline{\pi}$ ssi ils sont contenus dans le même bloc de π . La partition $\overline{\pi}$ est appelée le quotient de π par τ . Par exemple si :

$\pi = \{\overline{1,2,5,6} ; \overline{3,4,7}\}$ et $\tau = \{\overline{1,2} ; \overline{3,4} ; \overline{5,6} ; \overline{7}\}$

alors $\pi \geq \tau$ et le quotient de π par τ est le suivant :

$\overline{\pi} = \{\overline{B_1,B_3} ; \overline{B_2,B_4}\}$, où $B_1 = \{1,2\}$, $B_2 = \{3,4\}$,
$B_3 = \{5,6\}$, $B_4 = \{7\}$

Si $\pi_1 \geq \tau$ et $\pi_2 \geq \tau$, alors les quotients satisfont les relations suivantes :

$$\pi_1 \geq \pi_2 \text{ ssi } \overline{\pi}_1 \geq \overline{\pi}_2 \; ; \; \overline{\pi_1 \cdot \pi_2} = \overline{\pi}_1 \cdot \overline{\pi}_2$$

C'est-à-dire que les quotients se comportent exactement comme les partitions de départ.

Rappelons maintenant la définition d'un semitreillis. Un semitreillis est un couple $L = (X, \leq)$, où

(i) la relation "$\leq$" est un ordre partiel sur X
(ii) tout couple (x,y) d'éléments de X a une borne inférieure dans X, que l'on notera inf(x,y).

Il est facile à vérifier que :

(1) la relation "$\leq$" définie sur l'ensemble de partitions de S est un ordre partiel sur X
(2) $\inf(\pi_1, \pi_2) = \pi_1 \cdot \pi_2$

Il en résulte que l'ensemble de toutes les partitions de S muni de l'ordre partiel " $\leq$ " est un semitreillis. Nous allons rappeler maintenant une définition équivalente d'un semitreillis en termes de l'opération inf (notons que inf restreint à des couples devient une opération binaire). Quoique la définition que l'on a déjà vue plus haut soit plus simple, cette deuxième définition fait ressortir les aspects algébriques d'un semitreillis.

Un semitreillis est un couple $L = (X,.)$ où "." est une opération binaire sur X satisfaisant les axiomes suivants : pour tout élément x,y, et z dans X :

(i) $x.x = x$
(ii) $x.y = y.x$
iii) $x.(y.z) = (x.y).z$

Cette définition n'utilise pas explicitement le concept d'ordre partiel,mais il est facile de l'introduire en définissant :

$x \leq y$ ssi $x.y = x$

Alors on peut démontrer que $L = (X, \leq)$ satisfait la première définition et est donc un semitreillis avec $\inf(x,y) = x.y$. En effet :

$x.x = x$ implique que $x \leq x$ (réflexivité)
si $x \geq y$ et $y \geq x$,alors $y = x.y = y.x = x$ (symétrie)
$x \geq y$ et $y \geq z$ implique $x.y = y$ et $y.z = z$ et $x.z = x.(y.z)$.

De plus, comme $x.(x.y) = (x.x).y = x.y$, on obtient $x.y \leq x$ et de la même façon, $x.y \leq y$. D'autre part, si pour un z quelconque, on a : $z \leq x$ et $z \leq y$, alors $(x.y).z = x.(y.z) = x.z = z$, donc $z \leq x.y$. Il en résulte que : $\inf(x,y) = x.y$.

Notons que dans le semitreillis de partitions de S, l'ensemble de toutes les partitions a un inf qui est la partition $\{\{s\} \mid s \in S\}$, notée 0.

Deux partitions π_1 et π_2 sont <u>complémentaires</u> ssi $\pi_1 \cdot \pi_2 = 0$. Evidemment la partition 0 est le complément de toute partition π. D'autre part, soit π_1 et π_2 deux partitions complémentaires. Soit $\pi_1' \leq \pi_1$ et $\pi_2' \leq \pi_2$. On peut vérifier facilement qu'alors π_1' et π_2' sont complémentaires. Notons que la notion du complément joue un rôle important dans l'étude de la décomposition d'une base que l'on verra au chapitre 3.

2.2. DEFINITION DES OBJETS

Le problème que l'on se pose est le suivant : étant donné un ensemble S d'objets et un ensemble d'opérations U sur S, spécifier une vue par un couple (D,O) de sous-ensembles de U tels que :

(i) D définit les "objets" de la vue
(ii) D définit les opérations sur ces objets

Les opérations de U sont des applications de l'ensemble S dans lui-même. Les conditions que l'on pose sur ces opérations sont les suivantes :

(1) U contient l'identité sur S, que l'on notera i.
(2) Pour tout objet $t \in S$ et pour toute opération $u \in U$, il existe une opération $v \in U$ qui nous "ramène" à t, c'est-à-dire $vu(t) = t$. Dorénavant, on notera la composition de deux opérations par simple juxtaposition de leurs symboles.
(3) U est fermé sous composition.

Il s'agit de conditions naturelles, puisque (2) nous permet d' "effacer", alors que (3) signifie que si l'on peut appliquer deux opérations a et b, une à la fois, on peut aussi appliquer leur composition ab. L'ensemble de toutes les opérations sur S offre un exemple d'ensemble U qui satisfait les trois conditions ci-dessus.

On s'est déjà posé le problème de la spécification d'une vue, au chapitre précédent, où l'on a esquissé une solution à travers un exemple. Ce chapitre est consacré à un traitement formel, à partir des travaux présentés dans [27] . On commence, dans ce paragraphe, par la première partie du problème, c'est-à-dire la définition des "objets" de la vue. Remarquons tout de suite que ces "objets" n'interviennent nulle part explicitement dans notre modèle. Il est néanmoins utile de concevoir un "objet" comme un bloc d'une partition de S.

Définition 1 . Un sous-ensemble D de U est un ensemble de définition minimal ssi.

(i) $i \in D$

(ii) $\forall\, t \in S \quad \forall a \in D \quad \exists \quad b \in D$ tel que $ba(t) = t$

(iii) $\forall\, a \in D \quad \forall\, b \in D \quad ab \in D$ □

Notons que (ii) ci-dessus n'implique pas que b est l'inverse de a.La Figure 1 montre deux exemples d'ensembles de définition minimaux D et D'. Prenons l'exemple de l'opération $a \in D$. Sur l'objet $1 \in S$ il n'y a que l'opération b qui satisfasse la condition (ii), alors que sur l'objet $2 \in S$ il n'y a que l'opération c qui satisfasse cette même condition. Les deux théorèmes qui suivent fournissent la justification théorique (et intuitive) de la Définition 1.

Théorème 1. Soit D un ensemble de définition minimal. Soit "$\equiv$" la relation sur S définie par

$t \equiv t'$ ssi $\exists\, a \in D$ tel que $t = a(t')$

Alors "$\equiv$" est une relation d'équivalence sur S □

Démonstration : Il faut démontrer que "$\equiv$" est réflexive, symétrique et transitive.

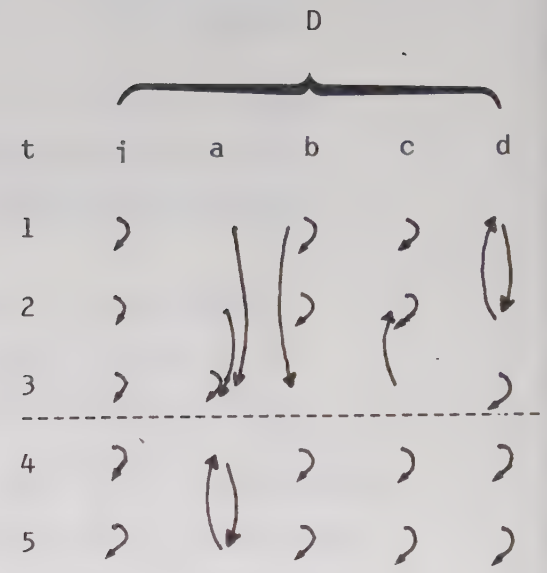
Réflexivité : Elle découle immédiatement du fait que $i \in D$.

Symétrie : Soit $t \equiv t'$. Il existe alors $a \in D$ tel que $t = a(t')$. La Définition 1 (ii) implique qu'il existe aussi $b \in G$ tel que $ba(t') = t'$. On a donc : $b(t) = ba(t') = t'$, c'est-à-dire $t' = b(t)$, d'où $t' \equiv t$ C.Q.F.D.

Transitivité : Soit $t \equiv t'$ et $t' \equiv t''$. Il existe alors a et b dans D tels que : $t = a(t')$ et $t' = b(t'')$, d'où $t = ab(t'')$. La Définition 1(iii) implique que $ab \in D$, donc $t \equiv t''$, C.Q.F.D.

D

	t	i(t)	a(t)	b(t)	c(t)	d(t)
S	1	1	3	1	1	2
	2	2	3	2	2	1
	3	3	3	1	2	3
	4	4	5	4	4	4
	5	5	4	5	5	5



D'

	t'	i'(t)	a'(t)	b'(t)	c'(t)
S'	1	1	2	1	3
	2	2	1	3	2
	3	3	3	2	1
	4	4	5	4	6
	5	5	4	6	5
	6	6	6	5	4

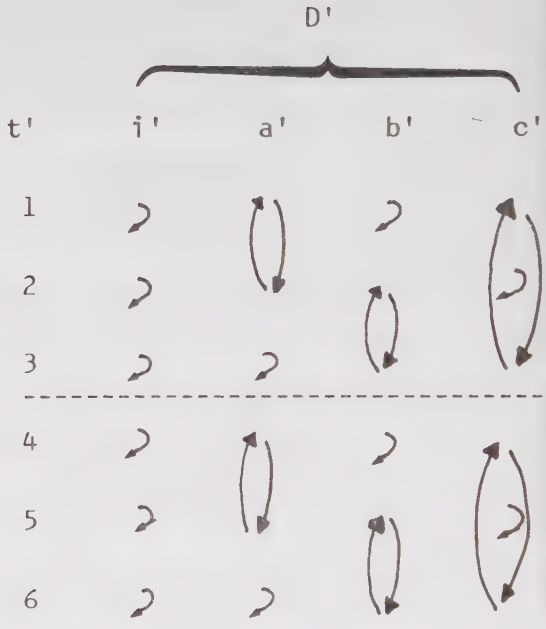


Figure 1 : Exemples d'ensembles de définition minimaux

Une autre façon de voir ce théorème est de dire que l'union des opérations de D est une relation d'équivalence sur S. Prenons le premier exemple de la Figure 1 et écrivons chaque opération de D sous forme d'un ensemble de couples. On obtient :

i = {(1,1), (2,2), (3,3), (4,4), (5,5)}
a = {(1,3), (2,3), (3,3), (4,5), (5,4)}
b = {(1,1), (2,2), (3,1), (4,4), (5,5)}
c = {(1,1), (2,2), (3,2), (4,4), (5,5)}
d = {(1,2), (2,1), (3,3), (4,4), (5,5)}

Si on forme l'union de ces ensembles, on obtient en effet une relation d'équivalence. Notons que dans le cas de D, on obtient cette même relation d'équivalence même si on enlève i. Ce n'est plus vrai dans le cas de D'.

Les classes d'équivalences de la relation " $\equiv$ " forment une partition de S, que l'on appelera la partition induite par D. Ainsi dans le premier exemple de la Figure 1, la partition induite par D a comme blocs :

$B_1 = \{1,2,3\}$ et $B_2 = \{4,5\}$

On peut constater que ces blocs sont invariants par les opérations de D, par exemple :

$a(B_1) \subset B_1$, $a(B_2) = B_2$, $d(B_1) = B_1$, etc.

Théorème 2. Soit D un ensemble de définition minimal. Soit π la partition de S induite par D. Alors

$\forall\, B \in \pi \quad \forall a \in D \quad a(B) \subset B$ □

Démonstration : Il suffit de démontrer que :

$\forall t \in B \quad \forall a \in D \qquad t \equiv a(t)$

La Définition 1(ii) implique qu'il existe $b \in D$ tel que $ba(t) = t$. Donc $b(t) = b(a(t))$, d'où $t \equiv a(t)$ C.Q.F.D.

Etant donné ce théorème, on appelera les blocs de la partition induite par D les objets de D.

Ayant établi, par les deux théorèmes précédents, le bien-fondé de la Définition 1, on pourrait se demander si cette définition est "minimale", c'est-à-dire, si les trois axiomes (i), (ii), et (iii) sont indépendants, au sens qu' aucun n'est impliqué par les deux autres. La Figure 2 démontre par des contre-exemples que ces axiomes sont, en effet, indépendants.

Revenons maintenant aux propriétés des ensembles de définition minimaux. La première question que l'on se pose est celle de l'unicité d'un ensemble de définition. Pour répondre à cette question, on a besoin des définitions suivantes.

Définition 2. Soit D un ensemble de définition minimal. Soit π la partition induite par D. Soit $u \in U$. On dit que u respecte π ssi $\forall B \in \pi \quad u(B) \subset B$ □
(notons que : $\forall u \in D$ u respecte π).

Définition 3. Soit D un ensemble de définition minimal. Soit π la partition induite par D. La fermeture de D, notée $I(\pi)$, est définie comme suit :
$I(\pi) = \{u \in U \mid u \text{ respecte } \pi\}$ □

t	I(t)	a(t)
1	1	1
2	2	2
3	3	1
4	4	2

(i) + (iii) $\not\Longrightarrow$ (ii)

t	I(t)	a(t)	b(t)	c(t)
1	1	4	3	1
2	2	3	4	2
3	3	3	1	2
4	4	4	2	1

(i) + (ii) $\not\Longrightarrow$ (iii)

t	a(t)	b(t)
1	2	1
2	2	1
3	3	3

(ii) + (iii) $\not\Longrightarrow$ (i)

Figure 2 : Contre-exemples démontrant la minimalité des conditions en Définition 1

Bien évidemment, si u respecte π, u ne relie pas deux objets différents de D. Ceci implique que l'ensemble $D \cup \{u\}$ induit la même partition de S que l'ensemble D. D'où la définition suivante :

Définition 4. Un sous-ensemble D' de U est un ensemble de définition ssi il existe un ensemble de définition minimal D tel que $D \subseteq D' \subseteq I(\pi)$ □

Au vue de ce résultat, la question qui s'impose est de savoir s'il existe un ensemble de définition minimal et unique. La réponse est négative, comme le montre l'exemple ci-dessous.

S	t	a(t)	b(t)	c(t)	d(t)	I(t)	e(t)
	1	1	3	1	2	1	1
	2	2	3	2	1	2	2
	3	1	3	2	3	3	1
	4	4	5	4	4	4	5
	5	5	4	5	5	5	4

D_1 et D_2 sont tous deux des ensembles de définition minimaux, en ce sens que, si on enlève une opération, le restant ne satisfait plus la Définition 1. Notons encore que l'intersection $D_1 \cap D_2$ n'est pas un ensemble de définition. Ceci implique que l'intersection de deux ensembles de définitions n'est pas forcément un ensemble de définition, même si les deux ensembles ont les mêmes objets. En ce qui concerne leur union, elle est un ensemble de définition si les deux ensembles ont les mêmes objets (ceci découle de la Définition 4). Ceci n'est plus vrai lorsque les deux ensembles ont des objets différents, comme le montre l'exemple ci-dessous.

			D_1	D_2
	t	a(t)	I(t)	b(t)
S	1	2	1	3
	2	1	2	2
	3	4	3	1
	4	3	4	4

Notons que les ensembles de définition qui induisent une même partition donnée π ont tous la même fermeture. Cette fermeture commune est l'ensemble de toutes les opérations de U qui respectent π .

La notion du complément d'une partition a une traduction intéressante en termes de fermetures, comme le montre le théorème suivant :

Théorème 3. Soit deux partitions π_1 et π_2 de l'ensemble S. Alors $\pi_1.\pi_2 = 0$ ssi $I(\pi_1) \cap I(\pi_2) = \{i\}$ □

Démonstration : Soit $\pi_1.\pi_2 = 0$. Evidemment, $i \in I(\pi_1) \cap I(\pi_2)$.

Supposons maintenant qu'il existe $a \in I(\pi_1) \cap I(\pi_2)$ tel que $a \neq i$. Ceci implique qu'il existe $s \in S$ tel que $s \neq a(s)$. D'autre part, $s \equiv a(s)(\pi_1)$ et $s \equiv a(s)(\pi_2)$ implique que $s \equiv a(s)(\pi_1.\pi_2)$, une contradiction, puisque $\pi_1.\pi_2 = 0$. Donc, $I(\pi_1) \cap I(\pi_2) = \{i\}$, C.Q.F.D.

Maintenant soit $I(\pi_1) \cap I(\pi_2) = \{i\}$. Supposons que $\pi_1.\pi_2 \neq 0$. Alors il existe $B \in \pi_1.\pi_2$, $s \in B$, et $t \in B$ tels que $s \neq t$. Soit l'opération $a : S \longrightarrow S$ telle que $a(s) = t$, $a(t) = s$, et $a(p) = p$ pour tout $p \in (S - \{s,t\})$. Bien évidemment $a \in I(\pi_1) \cap I(\pi_2)$ et, puisque $a \neq i$, on obtient une contradiction. Donc $\pi_1.\pi_2 = 0$, C.Q.F.D.

En conclusion, la différence entre un ensemble de définition D et sa fermeture $I(\pi)$ est la suivante :

- L'ensemble d'opérations D est une représentation (éventuellement minimale) de l'information nécessaire à la définition d'une partition π , c'est-à-dire à la définition des objets d'une vue.

- L'ensemble $I(\pi)$ est la "somme" de toutes les représentations possibles (par opérations) de l'information nécessaire à la définition de π . On pourrait voir $I(\pi)$ (et on le fera au chapitre suivant) comme l'ensemble de toutes les opérations de U qui peuvent être utilisées sans affecter les objets de D.

Enfin la partition π , qui est souvent implicite dans notre discours, forme le lien entre l'approche opérationnelle, proposée ici, et l'approche classique. En effet, ce sont les blocs de π qui, une fois "nommés" suivant un modèle (par exemple, le modèle relationnel) fournissent une réalisation de la vue.

2.3. DEFINITION DES OPERATIONS

Dans le paragraphe précédent on a considéré un ensemble S et un ensemble d'opérations U sur S, et on a examiné comment on pourrait définir les objets d'une vue, c'est-à-dire les blocs d'une partition π de S, par des opérations de U. Ayant défini les objets, il nous reste à définir les opérations "applicables" à ces objets, applicables au sens où elles transforment un objet en un autre et en un seul.

Définition 5. Soit π une partition de S. Une opération $u \in U$ est applicable à π ssi
$\forall B \in \pi \; \exists B' \in \pi$ tel que $u(B) \subseteq B'$ □

Etant donné un ensemble de définition D, on dira aussi qu'une opération $u \in U$ est applicable à D ssi u est applicable à la partition induite par D. On notera $A(\pi)$ l'ensemble de toutes les opérations applicables à π . Evidemment $I(\pi) \subseteq A(\pi)$.

Cette définition nous permet de spécifier une vue en se servant uniquement des opérations de U.

Définition 6. Une spécification sur U est un couple (D,O) de parties de U telles que :
D est un ensemble de définition
O est un ensemble d'opérations applicables sur D (c'est-à-dire une partie de $A(\pi)$) □

Selon cette définition, une vue sur U est spécifiée en deux étapes :

(i) Une première étape D qui contient des opérations de U qui définissent les objets et qui les laissent donc invariants.

(ii) Une deuxième étape O qui contient des opérations de U qui "transforment" les objets. Notons que O peut ne pas contenir D.

La figure 1 montre un exemple de deux spécifications, à partir d'un ensemble U d'opérations. Notons que, dans les deux cas, O contient une partie seulement de $A(\pi)$.

Pour mieux comprendre la notion d'une spécification, prenons un exemple dans le cadre du modèle relationnel (voir paragraphe 1.1.). Soit une base (R,S) définie par une seule variable relationnelle R et une dépendance fonctionnelle comme suit :

$R(A_1 : T_1 ; A_2 : T_2)$; C : $A_1 \longrightarrow A_2$ dans R

L'ensemble des valeurs S de cette base est défini par :

$S = \{R \mid C(R) = \text{vrai}\}$

Soit maintenant une vue (V, f(S)) de cette base définie par une seule variable relationnelle V et une transformation de base f comme suit :

$V(A_1 : T_1)$

$f : S \longrightarrow P(T_1)$ tel que $f(s) = R[A_1]$ $\forall s \in S$

($P(T_1)$ signifie l'ensemble de parties de T_1)

Nous allons maintenant spécifier cette même vue par des opérations sur la base. Soit U l'ensemble de toutes les applications de S dans S, c'est-à-dire de toutes les mises à jour permises sur la base. Les opérations suivantes appartiennent à U.

r(x,y,y') : remplacer le n-uplet (x,y) par le n-uplet (x,y')
s(x) : supprimer le n-uplet (x,y) $\forall y \in T_2$

A partir de ces opérations nous définissons maintenant D et O, deux sous-ensembles de U :

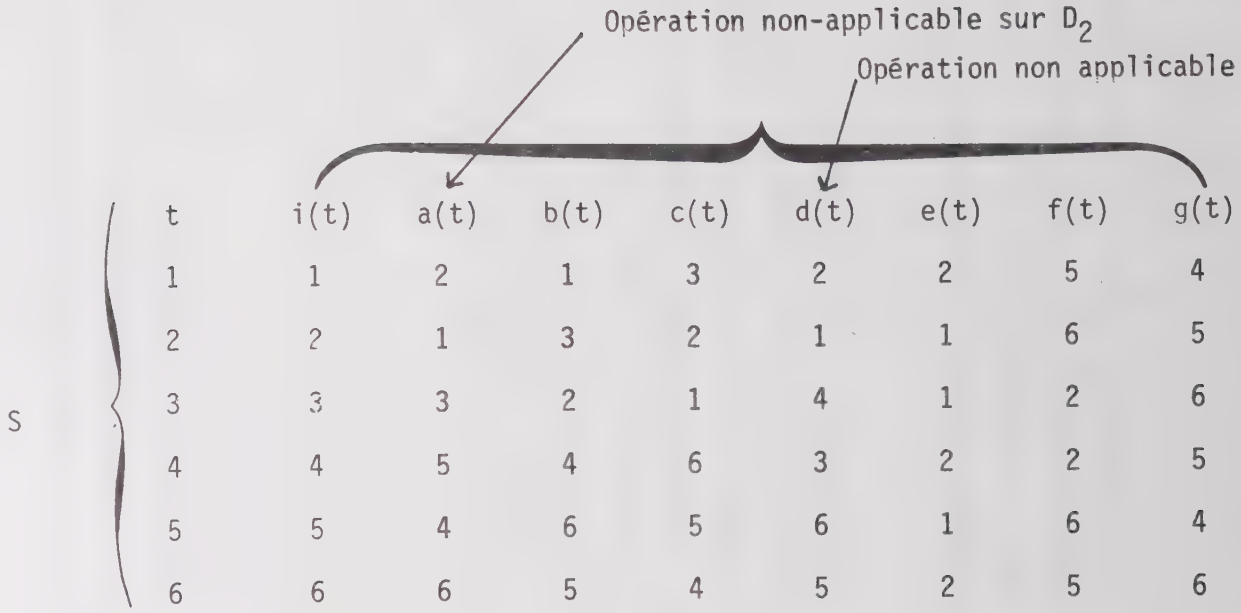


Figure 1 : Spécification de vues à partir d'un ensemble d'opérations U

Soit :

$D_o = \{r(x,y,y') \mid x \in T_1 , y \in T_2, y' \in T_2\}$

$D = D_o^*$ (c'est-à-dire, la fermeture de D_o sous composition)

$O = \{ s(x) \mid x \in T_1\}$

L'ensemble D satisfait les axiomes de la Définition 1, c'est donc un ensemble de définition minimal. Comme ses opérations laissent invariante la colonne A_1 de chaque relation R, cet ensemble "définit" la vue $(V, f(S))$. Selon le Théorème 1, deux valeurs (relations) de S sont équivalentes ssi elles ont la même projection sur A_1. Ainsi chaque opération de O fait passer d'une classe d'équivalence (bloc de partition) à une autre et une seule. Ceci implique que les opérations de O sont des opérations applicables sur la vue.

Bien évidemment, étant donnée une spécification (D,O), les opérations de O ne sont pas les opérations de la vue, mais ce sont plutôt des opérations de U traduisibles en opérations de la vue. Ainsi dans l'exemple ci-dessus, l'opération s(x) serait traduite par l'opération suivante de la vue:
$\bar{s}(x)$: supprimer x.
Nous avons déjà présenté un exemple d'une telle traduction au chapitre précédent (Figure 2), et nous aurons l'occasion d'y revenir par la suite.

Pour le moment, on voudrait insister sur le fait que cette spécification opérationnelle :

(1) d'une part, place sur le même plan toutes les vues définissables à partir de U, ce qui nous permettra de les comparer et d'étudier leurs interactions par des moyens très simples (c'est-à-dire par des opérations ensemblistes).

(2) d'autre part, et à la différence de l'approche classique, fait directement (et uniquement) appel au concept d'une opération sur la base, concept impliqué dans la presque totalité des problèmes de bases de données (processus de décomposition, formes normales, etc.).

Revenons maintenant à la définition de l'ensemble O. Etant donné un ensemble de définition D, pour former une spécification il faut définir un ensemble O d'opérations applicables sur D. C'est-à-dire qu'il faut définir un sous-ensemble de $A(\pi)$. Pour ce faire, on pourrait utiliser un prédicat, ou une contrainte sur $A(\pi)$. Bien sûr, cette contrainte doit être exprimée uniquement en termes d'opérations. Le théorème suivant fournit un exemple important d'une telle contrainte.

Théorème 4. Soit D un ensemble de définition minimal. Soit $u \in U$. Si $ua = au \quad \forall a \in D$ alors $u \in A(\pi)$ □

Démonstration : Soit π la partition induite par D. Il faut démontrer que :

$$\forall B \in \pi \ \exists B' \in \pi \text{ tel que } u(B) \subset B'$$

Soit $s \in B$ et supposons que $u(s) \in B'$ pour un $B' \in \pi$. Il suffit de démontrer que pour tout $t \in B$, $u(t) \in B'$. Tout d'abord, comme s et t appartiennent au même bloc de π, ils sont équivalents, au sens du Théorème 1. Donc, il existe $a \in D$ tel que $t = a(s)$. En utilisant l'hypothèse du théorème on obtient :

$$u(t) = u(a(s)) = a(u(s))$$

Comme $u(s) \in B'$, on obtient, par le Théorème 2, que $a(u(s)) \in B'$, c'est-à-dire $u(t) \in B'$, C.Q.F.D.

Ce théorème met en évidence l'importance capitale qu'a un ensemble de définition minimal pour l'approche opérationnelle des bases de données. En effet un ensemble de définition minimal nous permet de définir des opérations sur des objets que l'on ne "voit" pas ! Cela fournit une justification a posteriori de l'étude présentée dans le paragraphe précédent.

Quelques remarques importantes s'imposent maintenant au sujet du Théorème 4. Tout d'abord, on constate que plus l'ensemble de définition est petit, plus l'ensemble d'opérations applicables qui en résulte est grand. On a donc intérêt à choisir un ensemble de définition minimal si l'on veut un ensemble O maximal. Et comme on a, en général, plusieurs ensembles de définition minimaux qui induisent la même partition, on a intérêt à les considérer tous (et à réunir les ensembles d'opérations applicables qui en résultent), si l'on veut obtenir l'ensemble maximal définissable à partir de cette contrainte. Ensuite, il faut remarquer qu'il peut y avoir dans U (et même dans D) des opérations applicables sur D qui ne satisfont pas la condition du Théorème 4, comme le montre l'exemple ci-dessous.

D

S	t	I(t)	a(t)	u(t)	
	1	1	2	3	$u \in (U - I(\pi)$
	2	2	1	3	u est applicable mais
	3	3	4	4	$ua(1) \neq au(1)$
	4	4	3	3	

C'est-à-dire que cette contrainte ne suffit pas à définir toutes les opérations applicables sur une vue. Néanmoins, d'une part, elle fournit un exemple de contrainte non triviale sur les opérations applicables, et, d'autre part, elle définit des familles de bases très intéressantes. En effet, soit l'ensemble :

$C(D) = \{u \in U \mid ua = au \quad \forall a \in D\} \subseteq A(\pi)$

On pourrait définir une famille de bases comme suit :

$B = \{(D,O) \mid D$ un ensemble de définition minimal et $O = C(D)\}$

L'étude de cette famille pourrait bénéficier d'un appui algébrique assez puissant. Car l'ensemble C(D) est un être mathématique bien connu en algèbre. En effet, si on considère que U est fermé sous composition, alors C(D) est le "centralisateur" de D et, si D = U, alors C(U) est le "centre" de U. Ceci termine notre brève excursion algébrique.

Pour résumer, nous avons présenté la spécification opérationnelle d'une vue. Un exemple de contrainte sur U servant à la définition de l'ensemble O a aussi été introduit. Pour "réaliser" une vue à partir d'une telle spécification, il nous reste à "traduire" les opérations de O en opérations sur la partition π induite par D.

2.4. TRADUCTION DES OPERATIONS

Une spécification sur U a été définie comme un couple (D,O), où D est un ensemble de définition sur U et O un ensemble d'opérations applicables à D, c'est-à-dire un sous-ensemble de $A(\pi)$. Les opérations de l'ensemble $A(\pi)$ ne sont pas des opérations sur la vue, mais elles peuvent être traduites en opérations sur la vue.

Définition 7. Soit D un ensemble de définition sur U. Soit π la partition induite par D sur S. La traduction de $a \in A(\pi)$, notée a_π, est une opération (application) : $a_\pi : \pi \longrightarrow \pi$ définie comme suit :

$\forall B \in \pi \quad a_\pi(B) = B'$ ssi $a(B) \subset B'$, où $B' \in \pi$ □

Le fait que la traduction a soit une application bien définie, découle de la Définition 5. La Figure 2 montre la traduction des ensembles d'opérations O et O' de deux spécifications présentées dans la figure 1.

Etant données la spécification (D,O) et l'opération $a \in O$, un utilisateur de la vue appliquera a_π et non pas a. Une façon instructive d'interpréter l'opération a_π est la suivante. La spécification (D,O) définit une information partielle, ou incomplète sur la valeur actuelle $t \in S$. C'est-à-dire qu'on ne connait que le bloc $B \in \pi$ contenant t, ce qui représente une "incertitude" sur la valeur t. Si on applique a_π sur la vue, on obtient le bloc $B' \in \pi$ dans lequel se trouvera t, transformé par n'importe quelle opération de O ayant a_π comme traduction. Ainsi l'incertitude sur la valeur t reste "constante".

La traduction des opérations de $A(\pi)$ en opérations sur la partition π, induite par D, définit une application h de $A(\pi)$ dans l'ensemble de toutes les opérations sur π (comme le montre la Figure 2).

$D = \{i, a, b, c\}$ $D' = \{i, d\}$

$O = \{i, a, b, e, g\}$ $O' = \{i, d, e, f\}$

t	i	a	b	c	d	e	f	g
1	1	2	1	3	2	2	5	4
2	2	1	3	2	1	1	6	5
3	3	3	2	1	4	1	2	6
4	4	5	4	6	3	2	2	5
5	5	4	6	5	6	1	6	4
6	6	6	5		5	2	5	6

B	i_π	e_π	g_π
B_1	B_1	B_1	B_2
B_2	B_2	B_1	B_2

$B_1 = \{1,2,3\}$

$B_2 = \{4,5,6\}$

B'	$i_{\pi'}$	$e_{\pi'}$	$f_{\pi'}$
B'_1	B'_1	B'_1	B'_3
B'_2	B'_2	B'_1	B'_1
B'_3	B'_3	B'_1	B'_3

$B'_1 = \{1,2\}$

$B'_2 = \{3,4\}$

$B'_3 = \{5,6\}$

Figure 2 : Traduction des ensembles d'opérations O et O' de deux spécifications en Figure 1

Définition 8. Soit D un ensemble de définition sur U. Soit π la partition de S induite par D. Soit U_π l'ensemble de toutes les applications de π dans lui-même. Le traducteur de D est l'application suivante :

$h : A(\pi) \longrightarrow U_\pi$ tel que $h(a) = a_\pi \ \forall \ a \in A(\pi)$ □

Ainsi, dans le contexte de U, le traducteur de D définit l'ensemble des opérations permises sur π . Evidemment, étant donnée une spécification (D,O), l'ensemble des opérations permises sur π est h(O).

On se propose maintenant d'étudier les propriétés du traducteur h. On montre d'abord que l'ensemble $A(\pi)$ est fermé sous composition.

Théorème 5. Soit D un ensemble de définition sur U. Alors
si $a \in A(\pi)$ et si $b \in A(\pi)$, il en résulte que $ab \in A(\pi)$ □

Démonstration : Soit π la partition induite par D. Il suffit de démontrer que (voir Définition 5) :
$\forall \ B \in \pi \ \exists \ B' \in \pi$ tel que $ab(B) \subseteq B'$
Comme b est dans $A(\pi)$, il existe B" tel que $b(B) \subseteq B"$.
Comme a est dans $A(\pi)$, il existe B' tel que $a(B") \subseteq B'$.
On obtient donc,
$ab(B) = a(b(B)) \subseteq a(B") \subseteq B'$ C.Q.F.D.

On se sert maintenant de ce résultat pour démontrer que le traducteur h est, en fait un homomorphisme de l'ensemble $A(\pi)$ dans l'ensemble U_π.

Théorème 6 : Soit D un ensemble de définition. Soit π la partition induite par D. Soit $h : A(\pi) \longrightarrow U_\pi$ le traducteur de D. Alors
$\forall \ a \in A(\pi) \ \forall \ b \in A(\pi) \ \ h(ab) = h(a)h(b)$ □

Démonstration : Il suffit de démontrer que

$\forall\ B \in \pi \quad (ab)_\pi(B) = a_\pi(b_\pi(B))$

Soit : $b_\pi(B) = B'$ et $a_\pi(B') = B''$ (1)

D'où : $(a_\pi b_\pi)(B) = a_\pi(b_\pi(B)) = a_\pi(B') = B''$ (2)

Il suffit donc de démontrer que : $(ab)_\pi(B) = B''$

Notre hypothèse (1) et Définition 7 impliquent que

$b(B) \subseteq B' \quad$ et $\quad a(B') \subseteq B''$

D'où on obtient que

$(ab)(B) = a(b(B)) \subseteq a(B') \subseteq B''$

Ceci implique (Définition 7) que

$(ab)_\pi(B) = B''$ C.Q.F.D.

Evidemment, toutes les opérations de la fermeture $I(\pi)$ se traduisent par l'identité i_π de U_π. Et puisqu'il n'y a pas d'opérations de $A(\pi)$ en dehors de $I(\pi)$ avec cette propriété (Cf. Définition 3), on en déduit que $I(\pi)$ est le "noyau" de h, c'est-à-dire que $I(\pi) = h^{-1}(i_\pi)$. Ceci conduit à une propriété, à la fois simple et élégante de l'ensemble $I(\pi)$, exprimée par le Théorème suivant.

Théorème 7. Soit D un ensemble de définition. Soit $h: A(\pi) \longrightarrow U_\pi$ le traducteur de D. Soit x une opération quelconque dans l'image $h(A(\pi))$. Alors :

$\forall\ a \in h^{-1}(x) \quad aI(\pi) \triangleq \{ad \mid d \in I(\pi)\} \subseteq h^{-1}(x)$ □

Démonstration : Tout d'abord, $a \in h^{-1}(x)$ implique que $h(a) = x$.

Soit $d \in I(\pi)$. Alors $h(d) = i_\pi$ et, puisque h est un homomorphisme, on obtient $h(ad) = h(a)h(d) = xi_\pi = x$

Ceci implique que $ad \in h^{-1}(x)$, donc, $aI(\pi) \subseteq h^{-1}(x)$ C.Q.F.D.

Ce théorème nous permet donc à partir de n'importe quelle opération $a \in A(\pi)$, d'obtenir un ensemble $aI(\pi)$ d'opérations de $A(\pi)$, qui ont toutes la même traduction.

Notons que le Théorème 7 reste vrai même si on remplace l'ensemble $I(\pi)$ par n'importe quel sous-ensemble. Ceci suggère une nouvelle méthode de définition de l'ensemble O d'une spécification. En effet, étant donné un ensemble de définition D on pourrait définir un ensemble d'opérations O, et donc une spécification... "sur mesure" comme suit :

(i) Choisir un ensemble "générateur" $G \subseteq A(\pi)$

(ii) Choisir un ensemble "pilote" $P \subseteq I(\pi)$

(iii) Définir $O = \bigcup \{gP \mid g \in G\}$

(iv) (D,O) est la spécification engendrée par G et P

C'est une façon systématique de définir une spécification. Ainsi, si on veut que les opérations de G soient incluses dans O, on n'a qu'à inclure l'identité i dans P. De cette façon on "enrichit" tout simplement l'ensemble G, en "entourant" chaque opération $g \in G$ par des opérations "similaires", en ce sens qu'elles ont toutes la même traduction que g. Leur présence serait justifiée si, par exemple, la spécification (D,O) "coopérait" avec une autre spécification.

2.5. REALISATION D'UNE VUE

On a vu dans le paragraphe précédent qu'étant donnée une spécification (D,O), la traduction des opérations de O en opérations sur la vue se fait par le moyen d'une application $h : A(\pi) \longrightarrow U_\pi$. Cette application que l'on a appelée traducteur fait correspondre à chaque opération de O une opération de U_π (voir Figure 2). Evidemment, selon la représentation éventuelle des objets (c'est-à-dire des blocs de π), les "noms" dans U_π peuvent changer, ce qui implique un changement de "nom" pour le traducteur h. Ainsi, au lieu de l'application h que l'on a vue plus haut, on aurait une application $\hat{h}: A(\pi) \longrightarrow \hat{U}_\pi$. Ce qui reste inchangé, pendant ce changement de noms, c'est la partition induite par h sur $A(\pi)$, donc sur O. Chaque bloc de cette dernière partition donne lieu à une opération sur π et à une seule (à un nom près). On a donc intérêt à définir cette partition de O, indépendamment des noms. Ce que fait le théorème suivant dont la démonstration découle immédiatement de la Définition 7.

Théorème 8. Soit (D,O) une spécification. Soit π la partition de S induite par D. La relation " $\equiv$ " définie sur O par

$\forall\ a \in O\ \forall\ b \in O \quad a \equiv b$ ssi $a_\pi = b_\pi$

est une relation d'équivalence (on notera π' la partition sur O induite par " $\equiv$ ") □

Le Théorème 8 nous permet, à partir d'un ensemble de définition D, de définir non seulement un ensemble d'opérations O qui respectent π , mais aussi la partition π' induite par $\equiv$ sur O. Il implique aussi que le traducteur h' associé à π' ne soit qu'une restriction du traducteur h, défini sur l'ensemble $A(\pi)$ de toutes les opérations applicables à π .

Pour résumer, étant donnée une spécification (D,O),

(i) les blocs de la partition π de S induite par D, sont les objets de la vue

(ii) les blocs de la partition π' de O induite par π , sont les opérations de la vue.

En d'autres termes, il ne reste qu'à donner des noms, c'est-à-dire qu'à "représenter", suivant un modèle de données, les blocs des deux partitions π et π', pour "réaliser" la vue.

Définition 9. Soit (D,O) une spécification sur U. La réalisation de (D,O) est le couple (π , π') où
π est la partition de S induite par D
π' est la partition de O induite par π □

Evidemment, plusieurs spécifications peuvent avoir la même réalisation. Une spécification et sa réalisation définissent une vue.

Définition 10. Une vue de U est une spécification (D,O) sur U et sa réalisation (π , π') □

Evidemment la réalisation (π , π') "contient" la spécification, en ce sens que les liens qui existent entre D et O sont transformés en des liens entre π et π'. Nous utiliserons donc les termes : vue, spécification et réalisation d'une manière interchangeable. Ceci termine la définition opérationnelle d'une vue.

Pour résumer, la définition d'une vue comporte deux niveaux :

(i) la spécification opérationnelle

(ii) la réalisation suivant un modèle de données.

Cette séparation en deux niveaux, qui se rapproche de la notion d'un type abstrait de données, permet d'étudier les problèmes liés aux bases de données d'une manière simple et systématique comme l'on verra au chapitre suivant.

CHAPITRE 3 : APPLICATIONS

Dans ce chapitre nous présentons une étude de la décomposition d'une base par l'approche opérationnelle.

En particulier, nous définissons une structure de l'ensemble de décompositions d'une base qui nous permet de :

- Comparer deux décompositions
- Définir leur équivalence

Les résultats de cette étude sont ensuite appliqués à trois problèmes importants des bases de données :

- La gestion de vues d'une décomposition
- L'indépendance
- La définition d'une "vue première", notion en relation avec les formes normales.

3.1. LA DECOMPOSITION D'UNE BASE

On a vu la définition d'une vue sur U par une spécification (D,O) et sa réalisation (π,π'). Cette définition opérationnelle met en évidence les deux aspects d'une vue :

(i) l'aspect statique, c'est-à-dire, son contenu en information, représenté par la partition π,

(ii) l'aspect dynamique, c'est-à-dire la possibilité de transformer cette information, représenté par la partition π',

L'approche classique des bases de données utilise seulement l'aspect statique explicitement et l'aspect dynamique implicitement. C'est-à-dire que les opérations "légales" sur une base sont définies non pas explicitement, mais à travers des contraintes sur les données, telles que les dépendances fonctionnelles que l'on a vues au paragraphe 1.1., les dépendances multivaluées [18], etc. Etant donnés que les problèmes les plus importants des bases de données (telles que la décomposition, l'indépendance, les formes normales, etc.) impliquent explicitement les opérations sur la base, l'étude et même la compréhension de ces problèmes, à travers les contraintes deviennent difficiles. Nous pensons en effet que ces problèmes devraient être étudiés par une approche entièrement opérationnelle, et que l'on devrait transposer ensuite les résultats obtenus dans le cadre des contraintes spécifiques.

Après tout une contrainte sur les données n'est en effet qu'une contrainte sur les opérations que l'on peut faire sur ces données. C'est dans cet esprit que l'on va étudier dans ce chapitre quelques problèmes importants des bases de données, et montrer que l'on peut les énoncer d'une façon uniforme et générale, en dehors du cadre d'un modèle ou d'un type spécifique de contraintes. Ceci nous permettra de mieux comprendre ces problèmes et de mettre en évidence certains de leurs aspects qui ne sont pas liés aux contraintes. Dans ce paragraphe nous commençons par l'étude de la décomposition d'une base.

La définition d'une décomposition

Les utilisateurs d'une base de données n'ont pas tous les mêmes besoins. C'est-à-dire qu'

(i) un utilisateur peut ne pas avoir besoin de l'ensemble des informations de la base ,

(ii) un utilisateur peut ne pas avoir besoin de l'ensemble des opérations sur la base.

Ainsi le but de la décomposition d'une base est d'extraire, pour chaque utilisateur, les sous-ensembles d'informations et d'opérations qui lui sont nécessaires dans le cadre d'une application. Ce processus crée donc, pour chaque utilisateur, une vue de la base.

Définition 1. Une décomposition sur U est un ensemble fini de vues sur U □

Une seule vue v sur U étant aussi une décomposition sur U, on notera v la décomposition {v} .

Les vues d'une décomposition sont consultées et mises à jour par leurs utilisateurs au même titre que la base de départ. Mais alors qu'une requête sur une vue ne provoque pas de changement, une mise à jour sur une vue provoque des "effets de bords" sur les autres vues de la décomposition. Bien sûr pour qu'une décomposition soit cohérente, il faut pouvoir "calculer" les effets de bords provoqués par toutes les mises à jour de vues qui la forment. Prenc un exemple. Les Figures 1 et 2 montrent la décomposition d'une base (EMPLOYE, RAYON), en une vue EMPLOYE et une vue RAYON, spécifiées comme suit :

$$v_1 \begin{cases} D_1 = \{i,p,q,r\} \\ O_1 = \{i,p,q,r,k,l,n,t,u,v\} = A(\pi_1) \end{cases}$$

$$v_2 \begin{cases} D_2 = \{i,a,b,c,d\} \\ O_2 = \{i,r,w,z,a,b,c,d\} = A(\pi_2) \end{cases}$$

La Figure 3 montre la traduction de deux ensembles d'opérations O_1 et O_2 en opérations sur les vues. Si on applique l'opération $w_{\pi 2}$ sur v_2, il n'existe pas d'opération sur v_1 qui puisse calculer l'effet de $w_{\pi 2}$ sur v_2. Notamment, cet effet de bord de $w_{\pi 2}$ dépend de la valeur initiale de v_2. Par exemple, supposons que la valeur initiale de v_2 est {x,y} (voir Figure 2). Ceci implique que la valeur initiale de v_1 est {A,B} .
Après l'application de $w_{\pi 2}$, la valeur de v_2 devient {y} alors que celle de v_1 devient {A} ou {B} , selon que la valeur initiale de la base est la valeur n° 7 ou la valeur n° 8. Il n'existe donc pas d'opération sur v_1 qui puisse calculer la nouvelle valeur de v_1. Pour qu'une telle opération existe, il faut se restreindre à l'ensemble d'opérations applicables communes aux deux vues. C'est-à-dire qu'il faut se restreindre à l'ensemble $O_1 \cap O_2 = \{i,r\}$.

Définition 2. Une décomposition $d = \{v_1, v_2, \ldots, v_n\}$ sur U est cohérente ssi $O_1 = O_2 = \ldots = O_n$ □

Soit $X = O_1 = O_2 = \ldots = O_n$. Comme X, par ailleurs, doit être un sous-ensemble de $A(\pi_i)$, i = 1,2,...,n, cette définition implique que $X \subseteq A(\pi_1) \cap A(\pi_2) \cap \ldots \cap A(\pi_n)$. Par la suite, lorsque l'on parlera de décompositions, il s'agira toujours de décompositions cohérentes.

Une décomposition représente un ensemble d'informations extraites de la base de départ. Il est donc important de pouvoir comparer deux décompositions et de décider si elles sont équivalentes. Comme une décomposition est en quelque sorte la "somme" de ses vues, on a donc besoin d'un outil qui nous permette d'"ajouter" deux vues.

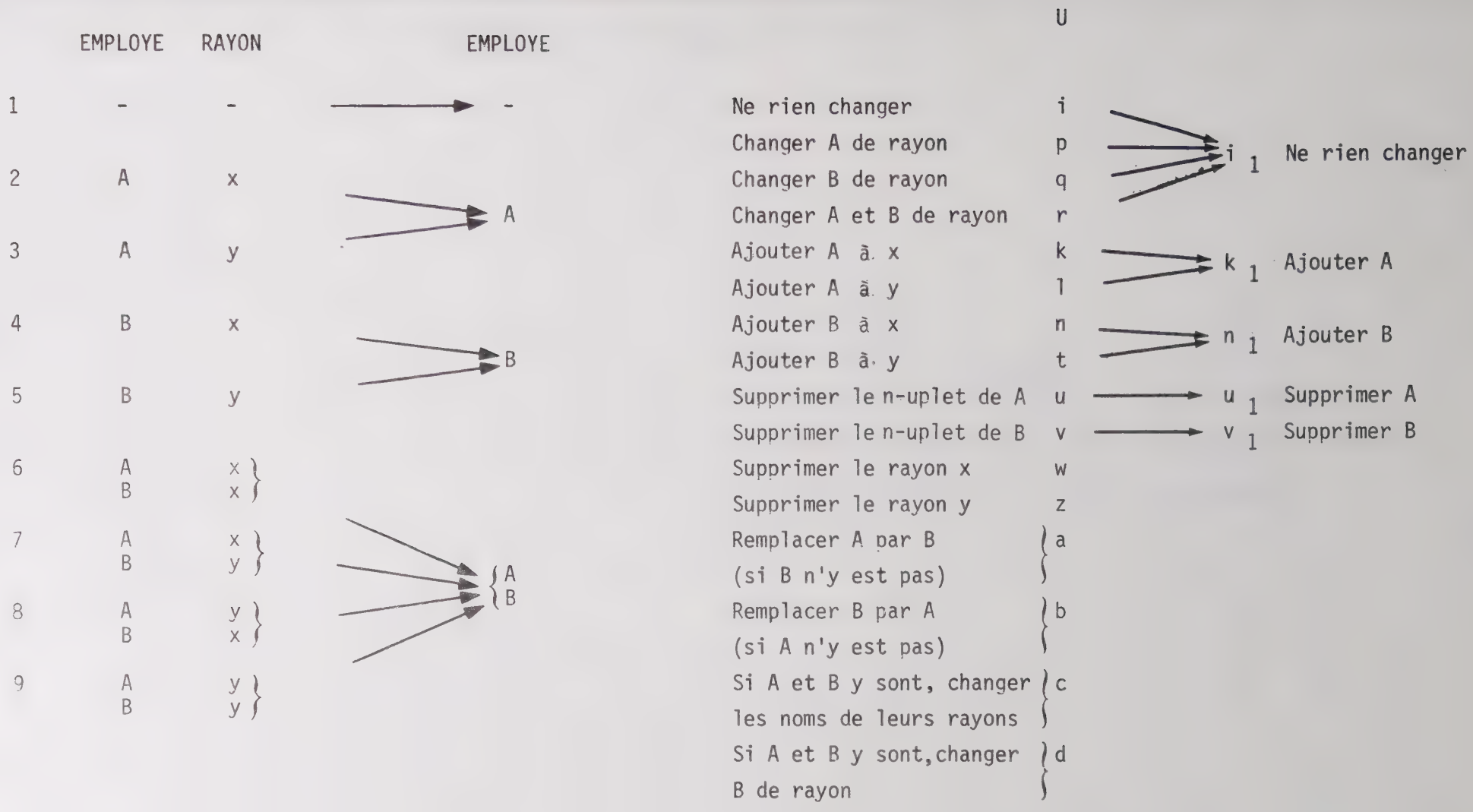


Figure 1 : La vue EMPLOYE et ses opérations

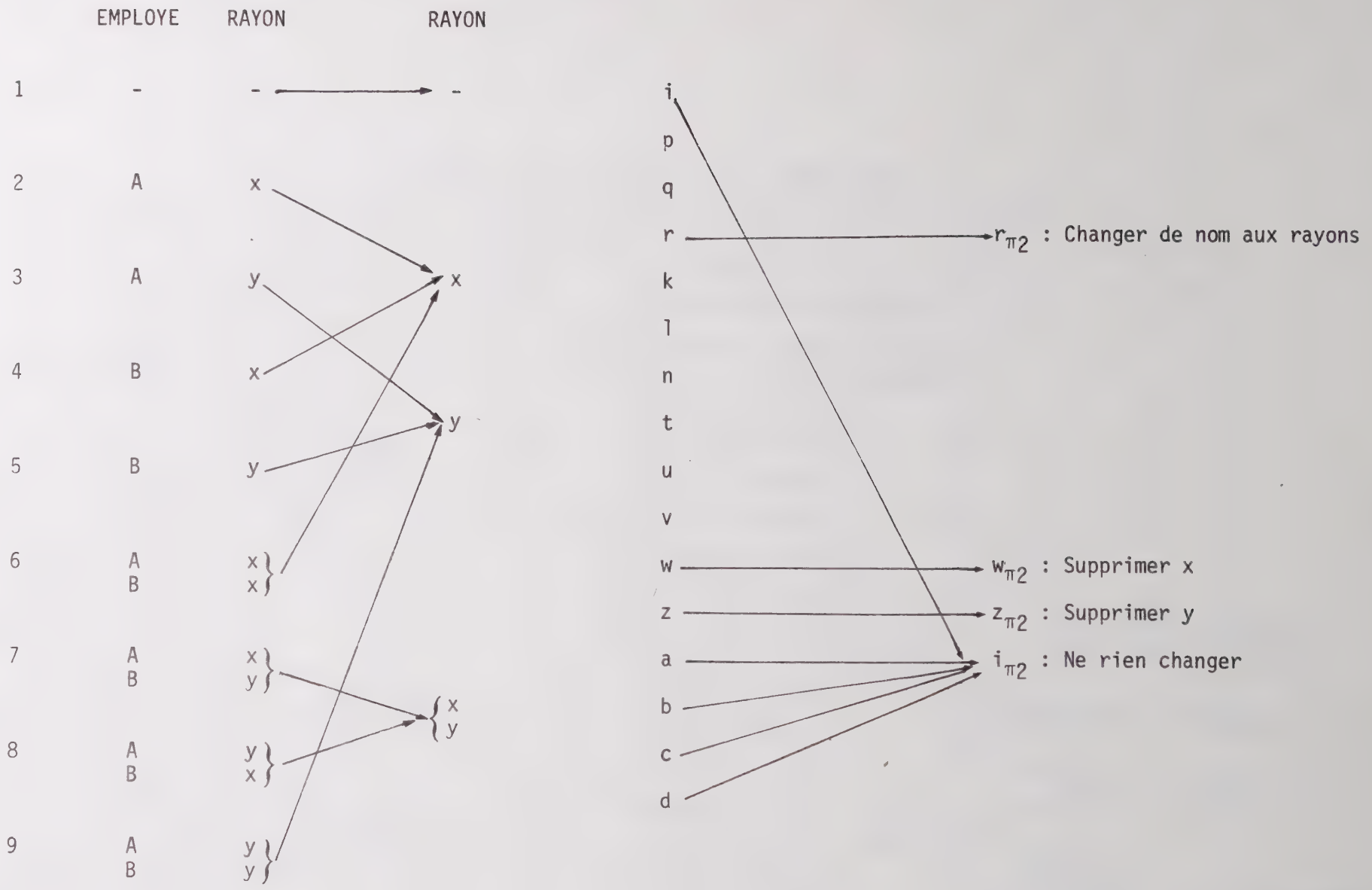


Figure 2 : La vue RAYON et ses opérations

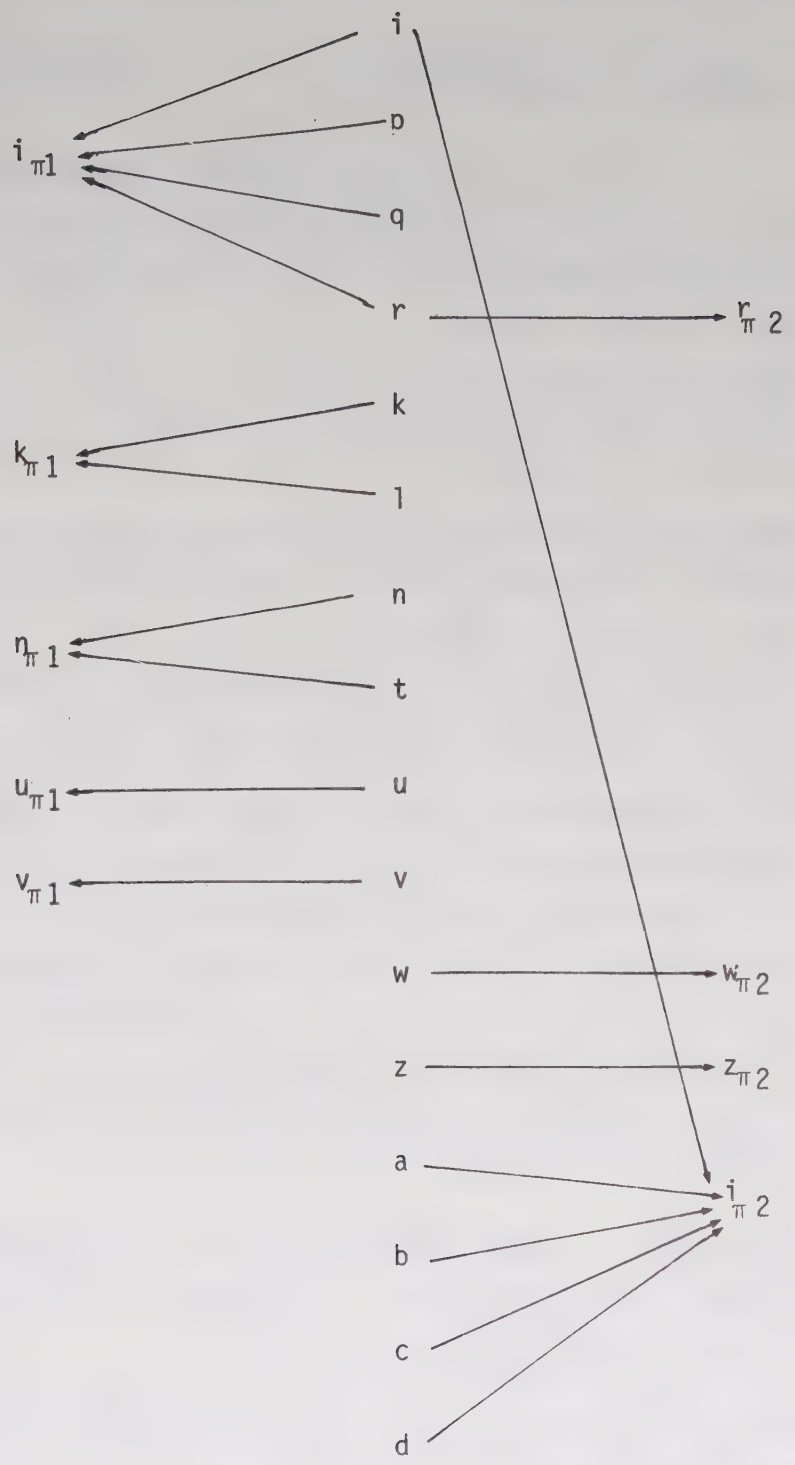


Figure 3 : La traduction des opérations applicables aux deux vues des figures 1 et 2

Structure de l'ensemble des vues

Nous nous proposons d'abord de comparer deux vues à travers leurs réalisations. Nous supposerons que les deux vues ont le même ensemble d'opérations. Soit X un sous-ensemble de U. Soit :
$R(X) = \{(\pi, \pi') \mid O = X\}$
C'est-à-dire que R(X) est l'ensemble de toutes les réalisations sur U dont l'ensemble d'opérations est X. Notons que R(X) contient la réalisation triviale (O,O').

Définition 3. Soit $r_1 = (\pi_1,\pi_1')$ et $r_2 = (\pi_2,\pi_2')$ deux réalisations de R(X). On dit que r_1 est inférieure ou égale à r_2, noté $r_1 \leq r_2$, ssi $\pi_1 \leq \pi_2$ et $\pi_1' \leq \pi_2'$ □
Il est facile de vérifier que la relation "≤ " définit un ordre partiel sur l'ensemble R(X). Quant à sa signification, on doit interpréter "$r_1 \leq r_2$" par "r_1 est plus fine que r_2". En d'autres termes $r_1 \leq r_2$ ssi les objets de r_1 affinent ceux de r_2 et les opérations de r_1 affinent celles de r_2. C'est une façon naturelle de comparer deux réalisations, puisqu'elle implique à la fois les objets et les opérations sur ces objets. Nous définissons maintenant le "produit" des deux réalisations r_1 et r_2. C'est une opération sur R(X) qui nous permet de les combiner pour en construire une qui soit plus fine que r_1 et r_2 et "optimale" en ce sens qu'elle utilise le minimum possible d'objets et d'opérations.

Définition 4. Soit $r_1 = (\pi_1,\pi_1')$ et $r_2 = (\pi_2,\pi_2')$ deux réalisations de R(X). Le produit de r_1 et r_2,noté $r_1.r_2$, est défini comme suit :
$r_1.r_2 = (\pi_1.\pi_2,\ \pi_1'.\pi_2')$ □
Notons que cette opération est réflexive, symétrique et associative. Nous allons montrer maintenant que l'ensemble R(X) est fermé sous l'opération "produit".

Lemme 1. Soit π_1 et π_2 deux partitions de S. Soit $a \in U$ une opération telle que $a \in A(\pi_1) \cap A(\pi_2)$. Alors $a \in A(\pi_1.\pi_2)$ □

Démonstration : Soit s et t deux points de S tels que
$s \equiv t(\pi_1)$ et $s \equiv t(\pi_2)$
Comme $a \in A(\pi_1) \cap A(\pi_2)$ on obtient
$a(s) \equiv a(t)(\pi_1)$ et $a(s) \equiv a(t)(\pi_2)$
Ce qui implique, par la définition de $\pi_1.\pi_2$, que $a(s) \equiv a(t)(\pi_1.\pi_2)$
Donc $a \in A(\pi_1.\pi_2)$, C.Q.F.D.

Lemme 2. Soit (π_1,π_1'), (π_2,π_2') deux réalisations de R(X), où $X \subseteq A(\pi_1) \cap A(\pi_2)$. Alors $(\pi_1.\pi_2)' = \pi_1'.\pi_2'$ □

Démonstration : Soit a et b deux opérations de X. Il faut démontrer que :
$a \equiv b((\pi_1.\pi_2)')$ ssi $a \equiv b(\pi_1'.\pi_2')$
Il suffit de démontrer que :
$a_{\pi_1.\pi_2} = b_{\pi_1.\pi_2}$ ssi $a_{\pi_1} = b_{\pi_1}$ et $a_{\pi_2} = b_{\pi_2}$
Supposons d'abord que $a_{\pi 1.\pi 2} = b_{\pi 1.\pi 2}$. Soit $B_1 \in \pi_1$. Il existe $B_2 \in \pi_2$ tel que $\emptyset \neq B_1 \cap B_2 \in \pi_1.\pi_2$. Par notre hypothèse, $\exists B_1' \in \pi_1$ $\exists B_2' \in \pi_2$ tels que $a(B_1 \cap B_2) \subseteq B_1' \cap B_2'$ et $b(B_1 \cap B_2) \subseteq B_1' \cap B_2'$
On en déduit que :
$a(B_1) \cap B_1' \neq \emptyset$ et $b(B_1) \cap B_1' \neq \emptyset$
et puisqu'il s'agit des blocs de π_1 on obtient
$a(B_1) \subseteq B_1'$ et $b(B_1) \subseteq B_1'$
d'où $a_{\pi 1}(B_1) = B_1' = b_{\pi 1}(B_1)$, C.Q.F.D.
De la même façon on peut démontrer que $a_{\pi 2} = b_{\pi 2}$
Supposons maintenant que $a_{\pi 1} = b_{\pi 1}$ et $a_{\pi 2} = b_{\pi 2}$. Soit $B_1 \cap B_2$ un bloc de $\pi_1.\pi_2$, où $B_1 \in \pi_1$ et $B_2 \in \pi_2$. Par notre hypothèse, $\exists B_1' \in \pi_1$ $\exists B_2' \in \pi_2$ tels que
$a(B_1) \subseteq B_1'$, $b(B_1) \subseteq B_1'$ et $a(B_2) \subseteq B_2'$, $b(B_2) \subseteq B_2'$

Il en résulte que

$a(B_1 \cap B_2) \subseteq a(B_1) \cap a(B_2) \subseteq B_1' \cap B_2'$ et

$b(B_1 \cap B_2) \subseteq b(B_1) \cap b(B_2) \subseteq B_1' \cap B_2'$

Donc, $a_{\pi 1.\pi 2}(B_1 \cap B_2) = B_1' \cap B_2' = b_{\pi 1.\pi 2}(B_1 \cap B_2)$, C.Q.F.D.

Théorème 1. Soit $r_1 = (\pi_1, \pi_1')$ et $r_2 = (\pi_2, \pi_2')$ deux réalisations de R(X). Alors

(i) $r_1.r_2 \in R(X)$

(ii) $r_1.r_2 = r_2$ ssi $r_1 \leq r_2$

(iii) $\inf(r_1,r_2) = r_1.r_2$ □

Démonstration : (i) Comme r_1 et r_2 appartiennent à R(X), le Lemme 1 implique que la réalisation

$r' = (\pi_1.\pi_2, (\pi_1.\pi_2)')$

appartienne à R(X). Le Lemme 2 implique alors que

$r' = (\pi_1.\pi_2, \pi_1'.\pi_2')$

C'est-à-dire $r_1.r_2 = r'$, donc $r_1.r_2$ appartient à R(X), C.Q.F.D.

(ii) Supposons que $r_1.r_2 = r_1$. Il en résulte que

$\pi_1.\pi_2 = \pi_1$ et $\pi_1'.\pi_2' = \pi_1'$

ce qui implique (voir section 2.1) que

$\pi_1 \leq \pi_2$ et $\pi_1' \leq \pi_2'$

Donc, $r_1 \leq r_2$ C.Q.F.D.

Supposons maintenant que $r_1 \leq r_2$. On a alors

$\pi_1 \leq \pi_2$ et $\pi_1' \leq \pi_2'$

et ceci implique que

$\pi_1.\pi_2 = \pi_1$ et $\pi_1'.\pi_2' = \pi_1'$

d'où $r_1.r_2 = r_1$, C.Q.F.D.

(iii) Par la commutativité et l'associativité du produit on obtient :

$(r_1.r_2).r_1 = r_1.(r_2.r_1) = r_1.(r_1.r_2) = (r_1.r_1).r_2 = r_1.r_2$

Donc $r_1.r_2 \leq r_1$ et, de la même façon, on obtient $r_1.r_2 \leq r_2$

Supposons maintenant que pour r appartenant à R(X) on ait

$r \leq r_1$ et $r \leq r_2$

On obtient alors

$r.(r_1.r_2) = (r.r_1).r_2 = r.r_2 = r$

d'où $r \leq r_1.r_2$

et ceci implique que $r_1.r_2 = \inf(r_1,r_2)$, C.Q.F.D.

Donc, étant données deux réalisations r_1 et r_2 de R(X) on sait :

- les comparer, par l'ordre partiel " $\leq$ ", pour en déduire la plus fine,
- les combiner par le produit ". ", pour en trouver une "optimale" qui raffine les deux à la fois

Ces deux opérations nous permettront de comparer deux décompositions et de décider si elles sont équivalentes.

Le contenu d'une décomposition

Soit $r_1 = (\pi_1,\pi'_1)$ et $r_2 = (\pi_2,\pi'_2)$, deux réalisations de R(X). On a vu que le couple $(\pi_1.\pi_2, \pi'_1.\pi'_2)$ est une réalisation de R(X) qui regroupe, sans redondance, les aspects statiques et dynamiques des deux réalisations r_1 et r_2. On peut donc dire que la réalisation $(\pi_1.\pi_2,\pi'_1.\pi'_2)$ est le "contenu" de la décomposition $d = \{r_1,r_2\}$. On appelera "décomposition sur X" toute décomposition dont les éléments sont des réalisations de R(X).

Définition 5. Soit $d = \{r_1,r_2,\ldots,r_n\}$ une décomposition sur X. Soit $r_i = (\pi_i,\pi_i')$, $i = 1,2,\ldots,n$. Le contenu de d, noté C(d), est une réalisation $r = (\pi,\pi')$ dans R(X) définie par

$\pi = \pi_1.\pi_2.\ldots.\pi_n$ et $\pi' = \pi'_1.\pi'_2 \ldots\ldots\pi'_n$ □

Le contenu d'une décomposition est donc une réalisation qui affine, sans redondance, toutes les réalisations de la décomposition. On pourrait donc comparer deux décompositions par leur contenu et décider si elles sont équivalentes.

Définition 6. Soit d_1 et d_2 deux décompositions sur X. On dit que d_1 est inférieure ou égale à d_2, noté $d_1 \leq d_2$, ssi $C(d_1) \leq C(d_2)$. On dit que d_1 est équivalente à d_2, notée $d_1 \equiv d_2$, ssi $C(d_1) = C(d_2)$ □

Si une décomposition d est équivalente à une réalisation r, on peut construire les objets et les opérations de r à partir des objets et des opérations de d, respectivement. On donnera à ces deux processus deux noms différents.

Définition 7. Soit $d = \{r_1, r_2, \ldots, r_n\}$ une décomposition, où $r_i = (\pi_i, \pi_i')$, $i = 1,2,\ldots,n$. Soit $r = (\pi, \pi')$ une réalisation.

(i) On dit que d préserve les objets par rapport à r, ssi $\pi_1 . \pi_2 \ldots . \pi_n = \pi$

(ii) On dit que d préserve les opérations par rapport à r, ssi $\pi_1' . \pi_2' \ldots . \pi_n' = \pi'$ □

Dans l'approche classique, la préservation des objets par rapport à r correspond à une décomposition de r qui "préserve l'information".

On pourrait se demander si la préservation des objets n'implique pas celle des opérations. Le théorème suivant montre que ceci est vrai en effet. Pour simplifier la démonstration de ce théorème nous supposerons que la décomposition n'a que deux éléments et qu'elle préserve les objets par rapport à la base de départ sur l'ensemble X.

Théorème 4. Soit $d = \{r_1, r_2\}$ une décomposition sur X. Si d préserve les objets par rapport à la base de départ, alors d préserve les opérations. □

Démonstration : Pour démontrer que $\pi_1' . \pi_2' = 0'$, il suffit de démontrer que :

$\forall\ a \varepsilon X\ \forall\ b \varepsilon X\ \ h_1(a) = h_1(b)$ et $h_2(a) = h_2(b) \Longrightarrow a = b$

Soit $t \varepsilon T$. Il suffit de démontrer que $a(t) = b(t)$. Par notre hypothèse, $\pi_1 . \pi_2 = 0$. Ceci implique que

$$\exists\ B_1 \varepsilon \pi_1\ \ \exists\ B_2 \varepsilon \pi_2 \text{ tels que } B_1 \cap B_2 = \{t\} \qquad (1)$$

D'autre part, $h_1(a) = h_1(b)$ implique que

$$\exists\ B_1' \varepsilon \pi_1 \text{ tel que } a(B_1) \subseteq B_1' \text{ et } b(B_1) \subseteq B_1' \qquad (2)$$

De même, $h_2(a) = h_2(b)$ implique que

$$\exists\ B_2' \varepsilon \pi_2 \text{ tel que } a(B_2) \subseteq B_2' \text{ et } b(B_2) \subseteq B_2' \qquad (3)$$

Par (2) et (3) on obtient

$$a(B_1) \cap a(B_2) \subseteq B_1' \cap B_2' \qquad (4)$$

$$b(B_1) \cap b(B_2) \subseteq B_1' \cap B_2' \qquad (5)$$

Par (1) et (4) on obtient

$$a(t) \varepsilon a(\{t\}) = a(B_1 \cap B_2) \subseteq a(B_1) \cap a(B_2) \subseteq B_1' \cap B_2' \qquad (6)$$

Par (1) et (5) on obtient

$$b(t) \varepsilon b(\{t\}) = b(B_1 \cap B_2) \subseteq b(B_1) \cap b(B_2) \subseteq B_1' \cap B_2' \qquad (7)$$

Ainsi (6) et (7) impliquent que

$$a(t) \varepsilon B_1' \cap B_2' \text{ et } b(t) \varepsilon B_1' \cap B_2' \qquad (8)$$

Comme $B_1' \cap B_2'$ appartient à $\pi_1 . \pi_2$, il contient un seul élément de T, et (8) implique alors que $a(t) = b(t)$, C.Q.F.D.

En conclusion, nous avons présenté la définition formelle de la décomposition d'une base. Nous avons vu aussi pourquoi une décomposition ne peut être cohérente que sur un ensemble d'opérations applicables à tous les éléments de la décomposition. La définition du contenu d'une décomposition nous a permis de comparer deux décompositions et de définir la notion d'équivalence. Enfin, nous avons démontré que la préservation des objets dans une décomposition implique celle des opérations et ceci nous permettra de mieux comprendre le processus de traduction dans une décomposition.

3.2. TRADUCTION DANS UNE DECOMPOSITION

Le but de ce paragraphe est de présenter, par l'approche opérationnelle, trois aspects importants de la traduction dans une décomposition, connus dans la littérature sous les noms de

- gestion des vues,
- décompositions indépendantes,
- formes normales.

Pour simplifier la présentation nous nous limiterons à une décomposition de la base de départ en deux éléments. Nous commençons par la définition du traducteur d'une décomposition.

Définition 8. Soit $d = \{ r_1, r_2 \}$ une décomposition sur l'ensemble $X = A(\pi_1) \cap A(\pi_2)$. Soit h_1 et h_2 les traducteurs de r_1 et r_2 respectivement. Le traducteur de d, noté h, est défini comme suit

$h : X \longrightarrow h_1(X) \times h_2(X)$ tel que $h(a) = (h_1(a), h_2(a)) \quad \forall\, a \in X$ □

Il est facile de vérifier que la partition de X induite par h est $\pi_1' . \pi_2'$. Ceci implique que si la décomposition préserve les opérations, son traducteur est injectif.

Gestion des vues

Les vues d'une base sont consultées et mises à jour par leurs utilisateurs au même titre que la base de départ. Deux cas peuvent se présenter en pratique.

(i) Seule la base est implantée alors que les vues sont "définies". Dans ce cas, une requête sur une vue peut être traduite facilement en une requête sur la base de départ et traitée comme une requête ordinaire. Pour ce qui est des mises à jour d'une vue, la situation est plus délicate. En effet, nous nous trouvons devant un problème de choix. Prenons l'exemple de la Figure 4, qui montre la traduction des opérations de l'en-

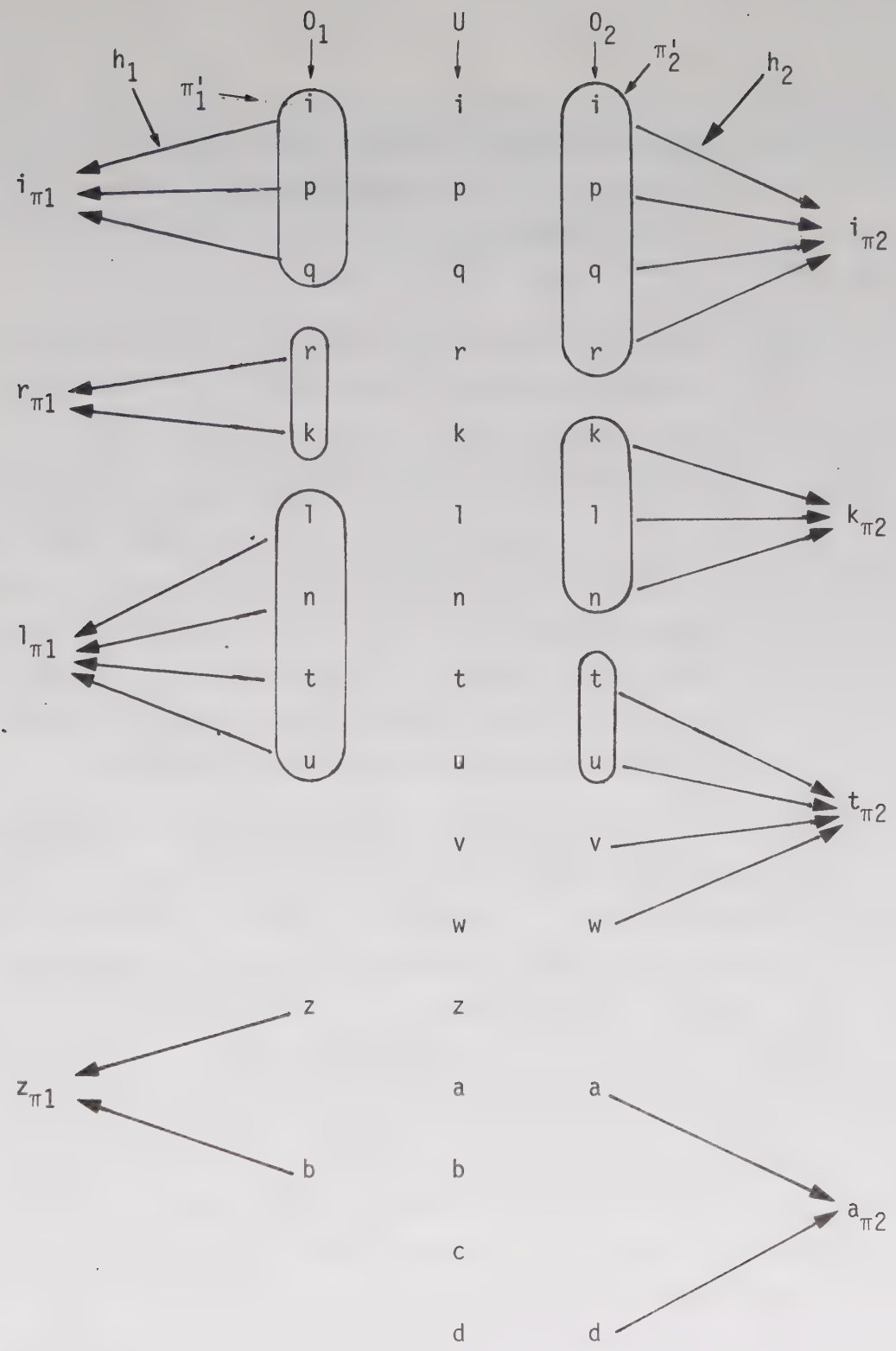


Figure 4 : Traduction des opérations de deux vues (hypothétiques) v_1 et v_2

semble $X = O_1 \cap O_2$. Etant donnée la mise à jour $k_{\pi 2}$ de la vue v_2, toutes les opérations de $h_2^{-1}(k_{\pi 2})$ peuvent servir de traduction.Il faut donc choisir parmi les opérations k,l, et n sur la base de départ.

(ii) Seules les vues sont implantées (cas d'une base répartie) et la base représente, en quelque sorte, l'"intégration" de ces vues. Dans ce cas, une requête sur une vue peut être traitée directement et ne présente aucun problème de traduction. Mais la mise à jour d'une vue aura, en général, des effets de bord sur les autres vues de la décomposition. Pour "calculer" ces effets de bord,on doit passer par une mise à jour sur la base de départ. Ainsi, en Figure 4, étant donnée la mise à jour $k_{\pi 2}$ de la vue v_2, l'effet de bord sur v_1 peut être soit $r_{\pi 1}$ soit $l_{\pi 1}$. Ceci dépend du choix fait parmi les opérations k,l, et n du bloc $h_2^{-1}(k_{\pi 2})$ de la partition π'_2 .

On voit donc que la gestion des vues d'une décomposition nécessite, pour chaque vue $v_i = (\pi_i, \pi'_i)$, la sélection d'une opération par bloc de π'_i . Le Théorème 4 suggère une méthode systématique de sélection. En effet, supposons qu'une vue "complémentaire" $\hat{v}_i = (\hat{\pi}_i, \hat{\pi}'_i)$ est associée à v_i.c'est-à-dire une vue telle que $\pi_i . \hat{\pi}_i = 0$. Ceci est toujours possible puisque toute partition a au moins un complément (voir paragraphe 2.1). Ainsi, on "force" une décomposition $d_i = \{v_i, \hat{v}_i\}$ qui préserve les objets et donc les opérations (Théorème 4), c'est-à-dire $\pi'_i . \hat{\pi}'_i = 0'$. Donc, si on fixe un bloc de $\hat{\pi}'_i$,celui-ci contient au maximum une opération de chaque bloc de π'_i. En d'autres termes, si on fixe une mise à jour $\hat{m}$ sur $\hat{v}_i$, celle-ci, associée à n'importe quelle mise à jour de v_i, détermine au maximum une opération (par bloc de π_i) sur la base. Ceci fournit une solution -partielle- au problème de la sélection.

Cette approche du problème de la sélection a été adoptée dans [4] où une solution est présentée dans le contexte du modèle relationnel. Cette solution utilise comme mise à jour $\hat{m}$ l'identité sur

$\hat{v}_i$, d'où son nom "traduction à complément constant". La vue complémentaire acquiert alors une interprétation sémantique intéressante, appelée dans [4] une "stratégie de mise à jour". Le choix de l'identité comme mise à jour $\hat{m}$ est justifié par le fait qu'elle associe à l'identité sur v_i l'identité sur la base (il découle du Théorème 3, Chapitre 2). Ceci correspond bien à la contrainte pratique qui n'exige pas de changement sur la base s'il n'y a pas de changement sur la vue.

Notons que le problème de la sélection une fois résolu, la gestion des mises à jour dans une décomposition peut se faire "en différé".

Retournons à l'exemple de la Figure 4. Supposons que les utilisateursdes vues demandent une suite de mises à jour, soit $r_{\pi 1}$, $k_{\pi 2}$, $r_{\pi 1}$, $t_{\pi 2}$, $l_{\pi 1}$, dans l'ordre de l'écriture, avant de poser une première requête. Supposons aussi que la suite correspondante de mises à jour sur la base est : k,k,k,t,t. Si s était la valeur de la base avant l'application du (premier) k, la réponse à la requête sera calculée à partir de la valeur ttkkk(s). On a évidemment intérêt à appliquer une seule fois l'opération ttkkk, au lieu d'appliquer les cinq opérations k,k,k,t,t, une à la fois.

Indépendance

Soit une décomposition $d = \{v_1, v_2\}$ sur l'ensemble $X = A(\pi_1) \cap A(\pi_2)$. Soit h_1 et h_2 les traducteurs de v_1 et v_2 respectivement. Soit h le traducteur de la décomposition. A chaque opération u de l'ensemble X, h associe un couple d'opérations $h_1(u)$ et $h_2(u)$ sur les vues. Cette association devient univoque lorsque la décomposition préserve les opérations. Dans le sens inverse, c'est-à-dire des vues vers la base,la situation est plus délicate. A partir d'un couple d'opérations (mises à jour) sur les vues, disons $x \in h_1(X)$ et $y \in h_2(X)$,on n'est pas toujours sûr de trouver une opération u sur la base telle que : $h_1(u) = x$ et $h_2(u) = y$ C'est-à-dire qu'en général on ne peut pas opérer indépendamment sur les deux vues. L'intérêt d'une telle indépendance est indiscutable.

surtout dans le cas d'une base répartie où chaque vue est sur un site différent.

Une décomposition est donc indépendante si $h(X) = h_1(X) \times h_2(X)$. De façon plus générale, on peut définir l'indépendance comme suit.

Définition 9. Soit $d = \{v_1, v_2\}$ une décomposition sur $X = A(\pi_1) \cap A(\pi_2)$. Soit $G_1 \subseteq A(\pi_1)$, $G_2 \subseteq A(\pi_2)$, et $H \subseteq X$. La décomposition d est indépendante sur G_1 et G_2 par rapport à H ssi $h_1(G_1) \times h_2(G_2) \subseteq h(H)$ □

En d'autres termes, d est indépendante ssi, étant donné n'importe quel couple d'opérations, $x \in h_1(G_1)$ et $y \in h_2(G_2)$, il existe une opération $u \in H$ telle que $h_1(u) = x$ et $h_2(u) = y$. Le problème qui se pose est de trouver de tels sous-ensembles G_1, G_2 et H. Le théorème suivant fournit un élément important de réponse.

Théorème 5. Soit une décomposition $d = \{v_1, v_2\}$ sur l'ensemble $X = A(\pi_1) \cap A(\pi_2)$. Soit $G = I(\pi_1) \cup I(\pi_2)$. Si $G \subseteq X$, alors
$h_1(G) \times h_2(G) \subseteq h(X)$
[C'est-à-dire que d est indépendante sur G par rapport à X]

Démonstration : On a

$$h_1(G) = h_1(I(\pi_1) \cup I(\pi_2)) = h_1(I(\pi_1)) \cup h_1(I(\pi_2)) = \{i_1\} \cup h_1(I(\pi_2))$$
$$h_2(G) = h_2(I(\pi_1) \cup I(\pi_2)) = h_2(I(\pi_1)) \cup h_2(I(\pi_2)) = h_2(I(\pi_2)) \cup \{i_2\}$$

Donc on obtient

$$h_1(G) \times h_2(G) = (\{i_1\} \cup h_1(I(\pi_2))) \times (h_2(I(\pi_1)) \cup \{i_2\})$$

Il faut démontrer maintenant que

$\forall x \in h_1(G) \quad \forall y \in h_2(G) \quad \exists u \in X$ tel que $h_1(u) = x$ et $h_2(u) = y$

Il y a quatre cas à considérer :

(i) Soit $x = i_1$ et $y \in h_2(I(\pi_1))$. Alors il existe $u \in I(\pi_1)$, tel que $h_2(u) = y$

Donc $h_1(u) = i_1 = x$ et $h_2(u) = y$ C.Q.F.D.

(ii) Soit $x = i_1$ et $y = i_2$. Comme $i \in G$ et $h_1(i) = i_1$ et $h_2(i) = i_2$, on a $h_1(i) = i_1 = x$ et $h_2(i) = i_2 = y$, C.Q.F.D.

(iii) Soit $x \in h_1(I(\pi_2))$ et $y \in h_2(I(\pi_1))$. Alors,

$\exists\ u \in I(\pi_2)$ tel que $h_1(u) = x$ (1)

$\exists\ w \in I(\pi_1)$ tel que $h_2(w) = y$ (2)

Comme $u \in G$ et $w \in G$, on a aussi $u \in X$ et $w \in X$. Donc, $uw \in X$. Par ailleurs, en utilisant (1) et (2) on obtient

$h_1(uw) = h_1(u)h_1(w) = x i_1 = x$

$h_2(uw) = h_2(u)h_2(w) = i_2 y = y$, C.Q.F.D.

(iv) Soit $x \in h_1(I(\pi_2))$ et $y = i_2$. Alors il existe $u \in I(\pi_2)$ tel que $h_1(u) = x$.

Donc $h_1(u) = x$ et $h_2(u) = i_2 = y$, C.Q.F.D.

On pourrait se demander quelle est la nature des opérations de l'ensemble $G = I(\pi_1) \cup I(\pi_2)$ pour lequel la décomposition est indépendante. Rappelons d'abord que toute opération de $I(\pi_1)$ laisse la vue v_1 inchangée. De même, toute opération de $I(\pi_2)$ laisse la vue v_2 inchangée. Donc toute opération de G laisse inchangée soit la vue v_1 soit la vue v_2 soit les deux vues à la fois. Ceci implique que toute opération de G laisse inchangée la "partie commune" des deux vues.

Pour mieux comprendre ce théorème, prenons un exemple. Soit une base relationnelle définie par une seule variable R et deux dépendances fonctionnelles comme suit.

R(NOM, ADRESSE, TELEPHONE)

F = { NOM ⟶ ADRESSE, NOM ⟶ TELEPHONE }

Soit aussi les deux vues suivantes obtenues par projections sur R

R_1(NOM, ADRESSE), F_1 = { NOM ⟶ ADRESSE }
R_2(NOM, TELEPHONE), F_2 = { NOM ⟶ TELEPHONE }

La Figure 5 montre cette décomposition en deux vues. Il est facile de voir qu'elle satisfait la condition du Théorème 5 . La "partie commune" de deux vues est la liste de noms.

Supposons maintenant que le nom "Spyratos" figure dans la base et considérons les opérations suivantes sur les vues.

x : Remplacer (Spyratos, Ottawa) par (Spyratos, Paris)
y : Remplacer (Spyratos, 2358786) par (Spyratos 3270037)

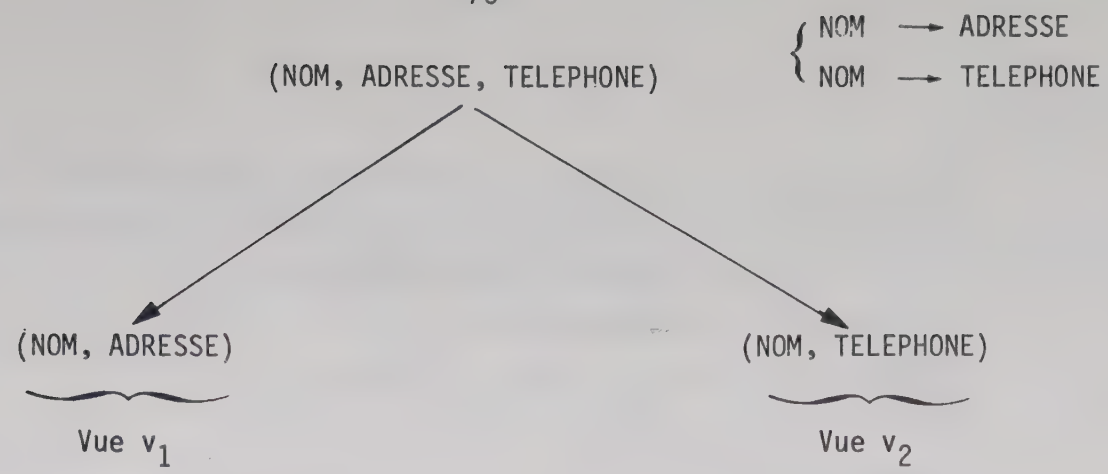
Comme la liste des noms ("partie commune" des deux vues) ne change pas, il existe une opération u sur la base qui se "projette" sur x et y. Cette opération est bien sûr la suivante :

u : Remplacer (Spyratos, Ottawa, 2358786) par (Spyratos, Paris, 3270037)

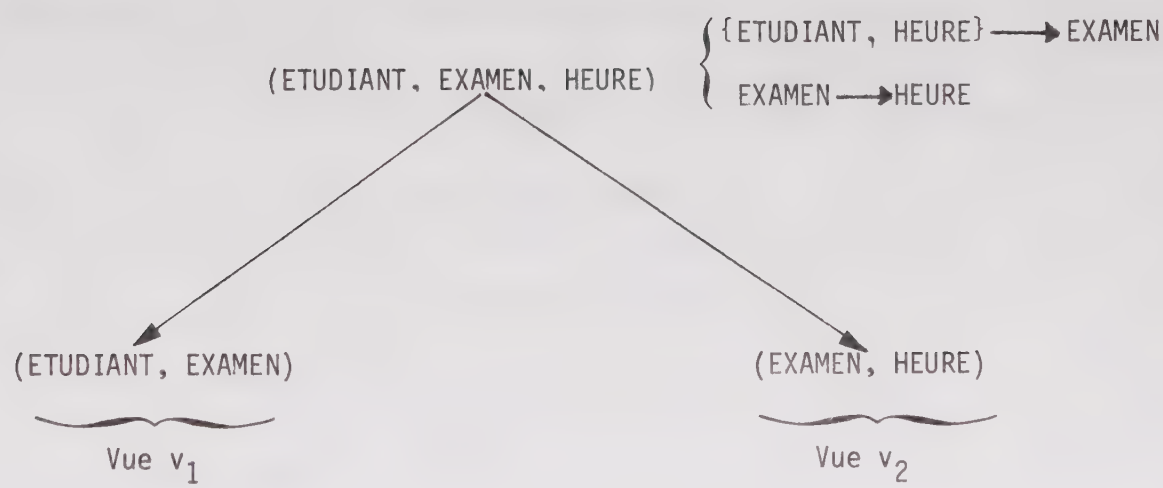
Donc l'intérêt d'une décomposition indépendante est qu' en assurant la cohérence de chaque vue,on assure la cohérence globale.

La deuxième décomposition de la Figure 5 ne satisfait pas la condition du Théorème 5. En effet, considérons l'opération :

u : Ajouter (Martin, Histoire, 9 heures), si (Histoire, 9 heures) figure dans la base et Martin est libre à 9 heures.



Décomposition faiblement indépendante



Décomposition non indépendante

Figure 5 : Exemples de décomposition

Cette opération est permise sur la base et elle appartient à $I(\pi_2)$ puisque elle ne change pas la vue v_2. Mais elle n'appartient pas à $A(\pi_1)$. En effet, supposons que la valeur de v_1 est l = {(Pierre, Histoire)} . Si l provient de la valeur s = {(Pierre, Histoire, 8 heures)} de la base, alors u ne change pas l. Mais si l provient de la valeur s' = {(Pierre, Histoire, 9 heures)} , u fait alors passer la vue de la valeur l à une nouvelle valeur l' : {(Pierre, Histoire), (Martin, Histoire)} . Ceci implique que l'opération u n'est pas applicable à la vue v_1. Nous concluons que :
$u \in I(\pi_1) \cup I(\pi_2)$ et $u \notin A(\pi_1) \cap A(\pi_2)$

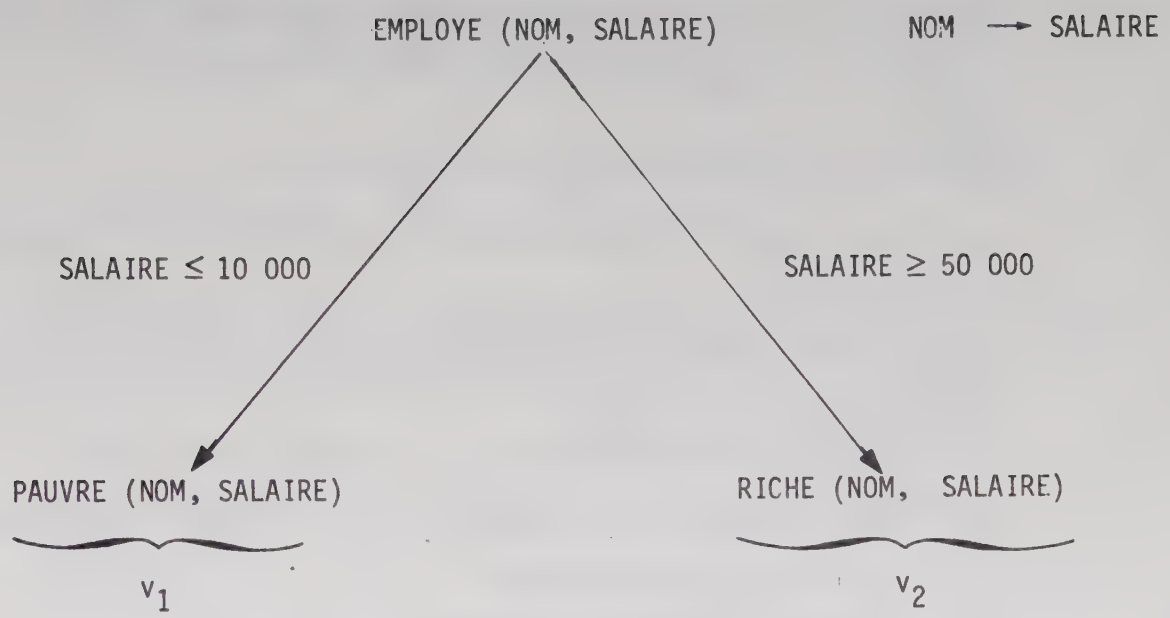
Donc la décomposition ne satisfait pas la condition du Théorème 5. Ceci implique qu'en assurant la cohérence de chaque vue dans cette décomposition,on ne sait pas si on assure la cohérence globale. Cette interprétation du Théorème 5 conduit à la définition suivante de l'indépendance "faible".

Définition 10. Une décomposition $d = \{v_1, v_2\}$ est faiblement indépendante ssi
$I(\pi_1) \cup I(\pi_2) \subseteq A(\pi_1) \cap A(\pi_2)$ □

Pour montrer que cette notion sort du cadre classique de la décomposition en plusieurs projections mais s'applique à d'autres types de décompositions, prenons maintenant l'exemple d'une base relationnelle et d'une décomposition définie non pas par projection mais par sélection (voir aussi Figure 6).

Base : EMPLOYE (NOM, SALAIRE), NOM ⟶ SALAIRE
Vue v_1 : PAUVRE (NOM, SALAIRE), tel que SALAIRE ≤ 10.000
Vue v_2 : RICHE (NOM, SALAIRE), tel que SALAIRE ≥ 50 000

Supposons que les employés de l'entreprise ont décidé de partager leurs primes de fêtes entre les pauvres jusqu'à concurrence de 1 000 Francs (le reste est ... réinvesti dans l'entreprise). Soit donc l'opération suivante sur la base.
u : Partager 50n francs entre les pauvres jusqu'à concurrence de 1 000 Francs (n est le nombre d'employés).



u : Soit n le nombre d'employés dans la base ; partager 50n francs entre les pauvres jusqu'à concurrence de 1 000 Francs.

$u \in I(\pi_2)$ mais $u \notin A(\pi_1)$

Figure 6 : Une décomposition non indépendante

Comme l'opération u laisse invariante la vue v_2, elle appartient à $I(\pi_2)$. D'autre part, comme la somme que chaque pauvre reçoit dépend du nombre d'employés n, l'opération u n'est pas applicable sur v_1. Donc,
$u \in I(\pi_1) \cup I(\pi_2)$ mais $u \notin A(\pi_1) \cap A(\pi_2)$

Ceci implique que la décomposition n'est pas faiblement indépendante.

Dans [5] nous avons étudié l'indépendance dans le cadre du modèle relationnel. La nouvelle définition que nous avons donnée ici (Définition 9) présente plusieurs avantages.

(i) Elle est naturelle, puisqu'elle ne fait appel qu'aux opérations sur la base de départ
(ii) Elle est indépendante du modèle
(iii) Elle est plus fine, puisque elle permet de modifier les ensembles impliqués. Nous avons pu obtenir ainsi une forme particulière de l'indépendance, que l'on a appelée "indépendance faible" (Définition 10),exprimée d'une façon simple et avec une interprétation naturelle
(iv) Elle est générale, puisque elle recouvre les définitions connues de l'indépendance.

Formes normales

Le processus de normalisation [14] a comme but d'éliminer certaines anomalies et redondances liées aux mises à jour. Prenons l'exemple d'une base relationnelle qui contient des informations concernant les fournisseurs d'une entreprise, disons, leurs noms, leurs adresses, les articles fournis et leurs prix. Au niveau de la conception, la question principale qui se pose est celle de l'organisation des informations. Commençons par une organisation regroupant toutes les informations dans une seule relation R comme suit :

R(FOURNISSEUR, ADRESSE, ARTICLE, PRIX)
FOURNISSEUR ⟶ ADRESSE
FOURNISSEUR, ARTICLE ⟶ PRIX

Cette organisation présente certaines anomalies et redondances pendant la mise à jour, notamment :

Anomalie d'insertion : On ne peut pas enregistrer le nom et l'adresse d'un fournisseur s'il ne fournit pas encore d'articles.

Anomalie de suppression : Si un fournisseur cesse de fournir des articles, on ne peut pas garder son nom et son adresse.

Redondance d'insertion : Lorsqu'on enregistre un nouvel article fourni par un fournisseur qui figure déjà dans la base, on est obligé d'enregistrer de nouveau le nom et l'adresse du fournisseur.

Redondance de suppression : Lorsqu'on supprime un fournisseur, s'il fournissait n articles différents, on est obligé de supprimer n fois son nom et son adresse.

Considérons maintenant une nouvelle organisation des informations en deux relations R_1 et R_2 comme suit :

R_1 (FOURNISSEUR, ADRESSE)
R_2 (FOURNISSEUR, ARTICLE, PRIX)

Il s'agit d'une décomposition de R qui préserve les objets, et qui n'a aucune des anomalies et redondances citées plus haut. Cette absence d'anomalies est dûe au fait que les deux relations R_1 et R_2 sont en "troisième forme normale" [6] . Comme son nom l'indique, ce n'est qu'une des nombreuses formes normales dans la littérature actuelle.

Le processus de normalisation a comme but l'extraction de toutes les "unités sémantiques" d'une organisation donnée

des informations. Naturellement ces "unités sémantiques" sont définies par les liens sémantiques qui existent entre les informations. Ainsi, dans l'exemple ci-dessus, les liens sémantiques entre
FOURNISSEUR, ADRESSE, ARTICLE, PRIX
sont exprimés par les deux dépendances fonctionnelles

FOURNISSEUR $\longrightarrow$ ADRESSE

{FOURNISSEUR, ARTICLE} $\longrightarrow$ PRIX

Ces dépendances fonctionnelles suggèrent les "unités sémantiques"
(FOURNISSEUR, ADRESSE), (FOURNISSEUR, ARTICLE, PRIX)
sur lesquelles on voudrait pouvoir opérer indépendamment.
Or un regroupement de toutes les informations en une seule relation
(FOURNISSEUR, ADRESSE, ARTICLE, PRIX)
réunit d'une façon artificielle deux "unités sémantiques" distinctes, d'où les anomalies citées plus haut. Donc cette relation n'est pas en forme normale parce qu'elle est décomposable en "unités sémantiques" distinctes. Ceci nous conduit à la définition suivante.

Définition 11. Soit π l'ensemble des objets d'une vue v. La vue v est première ssi

$\pi = \pi_1.\pi_2 \Rightarrow \pi = \pi_1$ ou $\pi = \pi_2$ □

Notons que le terme "première" a été choisi à cause de l'analogie évidente avec la définition d'un nombre premier.

A notre connaissance, les "anomalies" d'une décomposition n'ont jamais été définies formellement. La notion d'une vue première nous permet de mieux comprendre ces anomalies et de donner leur définition formelle. En effet, si une vue n'est pas première, alors

$\exists\, \pi_1, \pi_2$ tels que $\pi = \pi_1.\pi_2$, $\pi \neq \pi_1$, $\pi \neq \pi_2$

Ceci crée deux sources d'"anomalies":

Anomalie de redondance : le fait que $\pi \neq \pi_1$ et $\pi \neq \pi_2$ implique que $I(\pi) \neq I(\pi_1)$ et $I(\pi) \neq I(\pi_2)$. Donc il est possible que $\exists\, u \notin I(\pi_2)$ tel que $u \in I(\pi_1)$ et $u \in A(\pi_1) \cap A(\pi_2)$

Pour une telle opération u, on aura
$h_1(u) = i_1$ et $h_2(u) \neq i_2$

Dans l' exemple que l'on a vu plus haut, u peut être l'opération suivante
u : Enregistrer (Fournisseur = F, Adresse = A, Article = A',
Prix = P)
Si le fournisseur figure déjà dans la base, alors
$h_1(u)(R_1) = R_1 = i_1(R_1)$

Ceci exprime la partie redondante de l'opération u lorsqu'elle est appliquée sur la relation R. En effet, on est obligé d'enregistrer le nom et l'adresse d'un fournisseur qui est déjà enregistré. D'autre part, on a $h_2(u) \neq i_2$. Ceci exprime la partie non redondante de u.

Anomalie de traduction : le fait que $\pi \neq \pi_1$ et $\pi \neq \pi_2$ implique qu' il est possible d'avoir $x \in h_1(A(\pi_1))$ et $y \in h_2(A(\pi_2))$, sans qu'il y ait une opération $u \in A(\pi_1) \cap A(\pi_2)$ telle que $h_1(u) = x$ et $h_2(u) = y$. Par exemple, soit les opérations
x : Enregistrer (Fournisseur = F, Adresse = A) dans R_1
y : Ne rien enregistrer dans R_2
Si le fournisseur F ne figure pas dans R_2, il n'existe pas d'opération u sur la base qui se "projette" sur x et y.

Il est évident qu' une base qui n'est pas première peut avoir une décomposition en vues premières sans anomalies de redondance ni de traduction. Rappelons à ce propos qu' une décomposition indépendante n'a pas d'anomalies de traduction. Donc pour supprimer les anomalies de traduction, il suffit de définir judicieusement les ensembles d'opérations.

Ainsi dans l'exemple du fournisseur, la décomposition de R en R_1 et R_2 est faiblement indépendante. Supposons donc que

(i) Les opérations sur R_1 et R_2 sont restreintes à l'ensemble
$G = I(\pi_1) \cup I(\pi_2)$

(ii) Les opérations sur R sont restreintes à l'ensemble
$H = A(\pi_1) \cap A(\pi_2)$

Le Théorème 5 implique alors que $h_1(G) \times h_2(G) \subseteq h(H)$, c'est-à-dire que si on se restreint aux ensembles d'opérations G et H, on n'a plus d'anomalies de traduction. Néanmoins, il reste des anomalies de redondance.

En conclusion, nous avons étudié la notion d'une vue première qui nous a permis de mieux comprendre les anomalies de traduction et de redondance. Nous avons vu aussi que les anomalies de traduction d'une vue non première peuvent être enlevées par une décomposition indépendante.

CHAPITRE 4. CONCLUSION

En conclusion, nous esquissons une application possible de l'approche opérationnelle à la protection de l'information et nous discutons des parallèles qui existent entre machines abstraites et bases de données. Nous proposons enfin une représentation possible des contraintes d'intégrité par des opérations.

Nous avons présenté dans cette thèse une approche opérationnelle des bases de données et nous l'avons appliquée à quelques problèmes liés à la traduction dans une décomposition. Rappelons brièvement les éléments essentiels de cette approche.

Soit une base de départ définie par :

- un ensemble de valeurs S d'un type (référentiel) T
- un ensemble d'opérations U sur S

Une vue de cette base est définie à deux niveaux

Spécification opérationnelle : La vue est spécifiée par un couple (D,O) des sous-ensembles de U tels que

- D induit une partition de S(et ceci définit les objets de la vue)
- O induit des opérations sur la vue, c'est-à-dire que chaque opération de O se traduit en une opération sur la vue.

Réalisation : La vue est réalisée par un couple (π,π')de deux partitions telles que :

- π est la partition de S induite par D
- π' est la partition de O qui regroupe dans chaque bloc toutes les opérations qui ont la même traduction sur la vue.

Ce sont les blocs de ces deux partitions qui sont éventuellement "nommés" - suivant un modèle - pour réaliser la vue.

L'ensemble D ou la partition π représentent le contenu de l'information de la vue (aspect "statique"), alors que l'ensemble O ou la partition π' représentent la possibilité de transformer cette information (aspect "dynamique").

Nous avons appliqué l'approche opérationnelle à l'étude de la décomposition. Au cours de cette étude nous avons été amenés à définir le contenu d'une décomposition, et ceci nous a permis de comparer deux décompositions et de définir leur équivalence. Nous avons aussi appliqué l'approche opérationnelle à trois aspects importants de la traduction,

- traduction inverse de mises à jour d'une vue
- indépendance
- formes normales

Ces applications nous ont permis de voir que l'approche opérationnelle constitue un cadre naturel pour étudier le problème de la traduction. Nous allons maintenant suggérer d'autres applications possibles ainsi que les rapports qui existent entre l'approche opérationnelle présentée ici et la théorie des machines abstraites.

Protection de l'information

Chaque vue d'une base extrait une partie des informations, le "reste" de la base pouvant être considéré comme "protégé" ou caché à l'utilisateur. L'information partielle représentée par la vue reste inchangée tant que l'on utilise des opérations applicables à la vue. En effet, une opération applicable - au sens où on l'a vu dans cette thèse - ne fait que passer d'une valeur à une autre de l'information représentée par la vue. Le seul moyen qu'a l'utilisateur pour accroître cette information - c'est-à-dire sa connaissance de la base -, ce sont les opérations non applicables à la vue. Prenons un exemple. Soit une vue v_1 représentée ci-dessous par la partition $\pi_1 = \{B_1, B_2, B_3\}$ de l'ensemble des valeurs S de la base. L'opération u est une opération non applicable à la vue v_1, puisqu'elle fait passer de B_2 à B_1 ou B_3,

	s	u(s)	π_1	π_2
S	1	4	B_1	B_1
	2	2		
	3	2	B_2	B_{21}
	4	5		B_{22}
	5	4	B_3	B_3
	6	5		

selon la valeur de la base. Au niveau de la vue, le fait que $u(B_2)$ conduise a deux valeurs distinctes de la vue implique que B_2 contienne plus qu'une valeur de la base. Ceci nous conduit à une partition (vue) π_2 qui est plus fine que π_1, c'est-à-dire à une connaissance accrue de la base.

D'une façon plus générale, le problème qui se pose est le suivant. Etant donné une vue et un ensemble d'opérations sur la base - pas forcément applicables sur la vue -, trouver le maximum d'informations que l'on peut obtenir sur la base. Nous pensons que ce problème qui est directement lié à la protection de l'information, peut être traité par l'approche opérationnelle présentée dans cette thèse.

Machines abstraites et bases de données

Il y a quelques liens entre l'approche opérationnelle présentée ici et les machines abstraites que l'on voudrait signaler. Rappelons tout d'abord qu'une machine abstraite est un triplet (S,E,δ) où

- S est un ensemble fini d'états
- E est un ensemble fini de symboles d'entrée
- δ : $S \times E \rightarrow S$ est la fonction de transition

Chaque symbole d'entrée, une fois "lu" par la machine, fait passer de l'état courant à un "prochain état" de la machine. Cette transition est décrite par la fonction δ, qui est souvent représentée par une table. Par exemple, selon la "table de transition" ci-dessous, si la machine se trouve en l'état 3, après lecture du symbole d'entrée v, elle passe en l'état 1, c'est-à-dire $\delta(3,v) = 1$.

$S = \{1,2,3,4\}$

$E = \{u,v,w\}$

$\delta =$

	u	v	w
1	2	3	2
2	3	4	2
3	4	1	4
4	1	2	3

Il y a ici un parallèle évident avec les bases de données. En effet, nous pouvons concevoir une machine abstraite comme une base de données si nous imaginons que

- S est l'ensemble de valeurs d'une base
- E est un ensemble d'opérations (mises à jour) sur cette base
- $\delta(s,u) = u(s) \quad \forall\, s \in S \quad \forall\, u \in E$

Cette façon de concevoir une machine abstraite comme une base de données, et vice-versa, nous permet aussi de voir un certain nombre de problèmes des machines abstraites, tels que la redondance, la décomposition etc., comme problèmes des bases de données. Ceci nous permet de penser qu'une étude approfondie des liens qui existent entre machines abstraites et bases de données pourrait conduire à des résultats intéressants

Opérations et contraintes d'intégrité

L'approche opérationnelle présentée dans cette thèse fait appel à une base de départ définie par

- un ensemble de valeurs S qui est un sous-ensemble d'un referentiel T
- un ensemble d'opérations (mises à jour) U sur S.

Est-il possible de définir S et U par des opérations sur T ? La réponse à cette question permettrait en particulier la définition de S par des opérations sur T, c'est-à-dire la définition de contraintes d'intégrité - au sens de l'approche classique - par des opérations sur T. Nous esquissons ici une réponse possible à cette question. Soit

- A(T), l'ensemble de toutes les applications de T dans T
- C, un sous-ensemble de A(T), fermé sous composition, que l'on appelera la contrainte
- t_0, une valeur fixe de T que l'on appelera la base vide

La contrainte C définit un sous ensemble S de T et un sous-ensemble U' de A(T) comme suit

$S = \{a(t_o) \mid a \in C\}$
$U' = \{u \in A(T) \mid u(S) \subseteq S\}$

On pourrait interpréter S comme l'ensemble des valeurs accessibles par des applications de C, à partir de la base vide. Soit maintenant U l'ensemble des restrictions sur S de toutes les applications de U'. On pourrait interpréter U comme l'ensemble de toutes les mises à jour de S. Ainsi le couple (S,U) définit une base de départ, au sens de l'approche opérationnelle.

Une vue de cette base est définie par une partition π de S, comme l'on a vu dans cette thèse. Est-ce possible de définir π à partir d'une partition $\hat{\pi}$ de T et d'une contrainte $\hat{C}$ sur $\hat{\pi}$? Si oui, alors $\hat{C}$ est la "traduction" de C sur $\hat{\pi}$. Dans l'exemple

BASE : (NOM,ADRESSE,TELEPHONE)
F1 : NOM → ADRESSE, F2 : NOM → TELEPHONE

VUE : (NOM,ADRESSE)

la contrainte C correspond à : F1 ∧ F2, qui porte sur les valeurs de la base, alors que sa "traduction" $\hat{C}$ correspond à F1, qui porte sur les valeurs de la vue.

Nous pensons qu'une étude approfondie de ce processus de traduction des contraintes permettrait de transposer tous les résultats de l'approche opérationnelle dans le contexte de l'approche classique.

REFERENCES

[1] AHO A.V., BEERI C. and ULLMAN J.D. (1979). "The theory of joins in relational databases", ACM Trans. Database Systems, 4, pp. 297-314

[2] ANSI/X3/SPARC, The ANSI/X3/SPARC framework, D. Tsichritzis and A. Klug, eds., AFIPS Press,(1978).

[3] ARMSTRONG W.W. (1974). "Dependency structures of data bases relationships", Proc. IFIP 74, North-Holland, Amsterdam, pp. 580-583.

[4] BANCILHON F., SPYRATOS N. (1981). "Update semantics of relational views", ACM Trans. Data Base Systems. December.

[5] BANCILHON F., SPYRATOS N. (1981). "Independent components of data bases", Proceedings VLDB7. Cannes, France. Septembre.

[6] BEERI C., BERNSTEIN P. and GOODMAN N. (1978). "A sophisticate's introduction to database normalization theory", Proc. 4th Int. Conf. on Very Large Data Bases, West Berlin, pp. 113-124.

[7] BERNSTEIN P.A. (1976). "Synthesizing third normal form relations from functional dependencies", ACM Trans. Database Systems, 1, pp. 277.298.

[8] BEERI C., MENDELZON A.O., SAGIV Y. and ULLMAN J.D. (1979). "Equivalence of relational database schemes", in Proceedings, 11th ACM Symp. on Theory of Computing, pp. 319-329.

[9] BEERI C. and RISSANEN J. (1980)."Faithful Representation of Relational Database Schemes", IBM Research Report RJ2722.

[10] BERNSTEIN P.A. (1976). "Synthesizing third normal form relations from functional dependencies", TODS 1:4, pp. 277-298.

[11] BISKUP J., DAYAL U. et BERNSTEIN P.A. (1979). "Synthesizing independant database schemas", ACM/SIGMOD International Symposium on Management of Data, pp. 143-152.

[12] CHAMBERLAIN D.D., GRAY J.N. et TRAIGER D.D. (1975). "Views, authorization and locking in a relational data base system", Proceedings of AFIPS NCC, Vol. 44.

[13] CODD E.F. (1970). "A relational model for large shared data banks" Comm. ACM 13:6, pp. 377-387.

[14] CODD E.F. (1972). "Further normalization of the data base relational model, in "Data Base Systems" (R. Rustin, Ed.), Prentice-Hall, Englewood Cliffs, N.J.

[15] DAYAL U. et BERNSTEIN P.A. (1978). "On the updatability of relational views". Procedings of the 4th International Conference on very large data bases, West Berlin.

[16] DELOBEL C. et CASEY R.C. (1972). "Decomposition of a database and the theory of Boolean switching functions". IBM J. Res. 17 : 5, pp. 370-386.

[17] DELOBEL C. (1978). "Normalization and hierarchical dependencies in the relational data model, TODS 3:3, pp. 201-203.

[18] FAGIN R. (1977). "Multivalued dependencies and a new normal form for relational databases, TODS 2:3, pp. 262-278.

[19] FAGIN R. (1979). "Normal forms and relational database operators", in "Proceedings, ACM-SIGMOD Intl. Conf. on Management of Data", pp. 153-160.

[20] HARTMANIS J., STEARNS R.E. (1966). "Algebraic structure theory of sequential machines". Prentice-Hall.

[21] MENDELZON A.O. and MAIER D. (1979). "Generalized mutual dependencies and the decomposition of database relations", in "Proceedings, 5th Intl. Conf. on Very Large Databases", pp. 75-82.

[22] NICOLAS J.M. (1978). "Mutual dependencies and some results on updecomposable relations". Proc. ACM Intl. Conf. on Very Large Data Bases, pp. 360-367.

[23] PAOLINI P. et PELEGATTI G. (1977). "Formal definition of mappings in a data base". ACM/SIGMOD, Proceedings of the Int. Conf. on Management of Data, August, pp. 40-46.

[24] RISSANEN J. (1977). "Independent components of relations". TODS 2:4, pp. 317-325.

[25] RISSANEN J. (1978). "Theory of relations for databases-A tutorial survey". Proceedings, Symp. on Mathematical Foundations of Computer Science, Zakopane, Poland, Springer-Verlag, Berlin/New York.

[26] SEVCIK K.C., FURTADO A.L. (1978). "Complete and Compatible Sets of Update Operations". Proc. ICMOD 78, Milan, Italie, Juin, pp. 247-260.

[27] SPYRATOS N. (1980). "Translation structures of relational views" Proceedings of the Sixth International Conference on Very Large Data Bases, Montréal, October.

[28] SPYRATOS N., BANCILHON F. (1979). "The abstraction process in the relational Model", Rapport de Recherche n° 342, INRIA-LABORIA.

[29] ULLMAN J.D. (1980). "Principles of Database Systems", Computer Science Press, Potomac, Md.

[30] WEBER H. (1976). "The D-graph model of large shared data bases : A representation of integrity constraints and views as abstract data types", IBM research report, RJ 1875 (27024), 11/22/76.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique

Dépôt légal : 301181 / 350

ISBN 2-7261-0278-6